Introduction to matplotlib

In [1]:

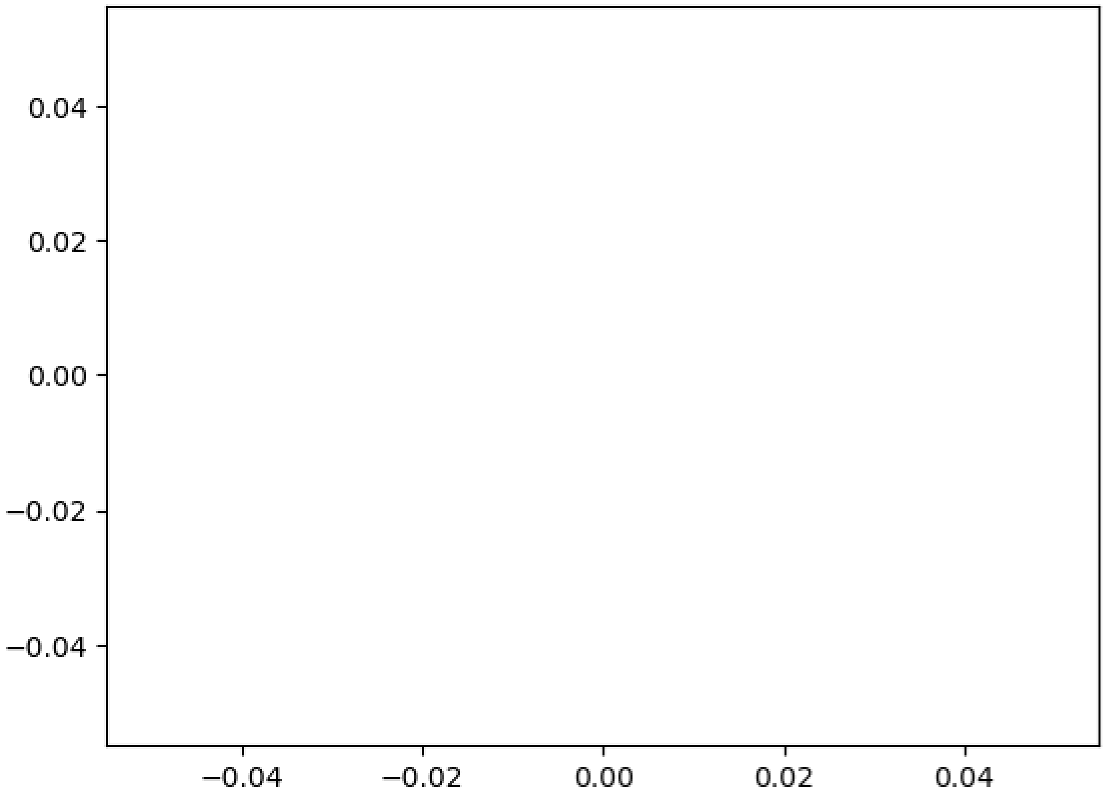
```
%matplotlib inline
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
```

In [2]:

```
# simplest way to create a plot
plt.plot()
```

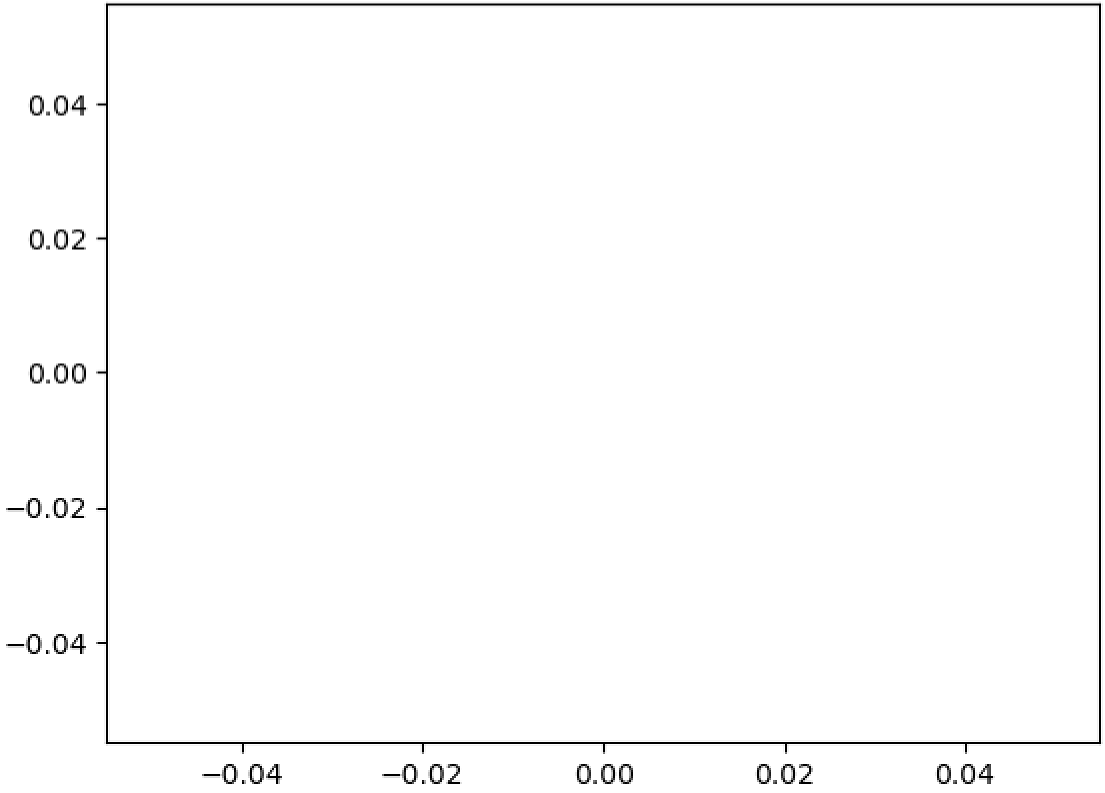
Out[2]:

```
[]
```

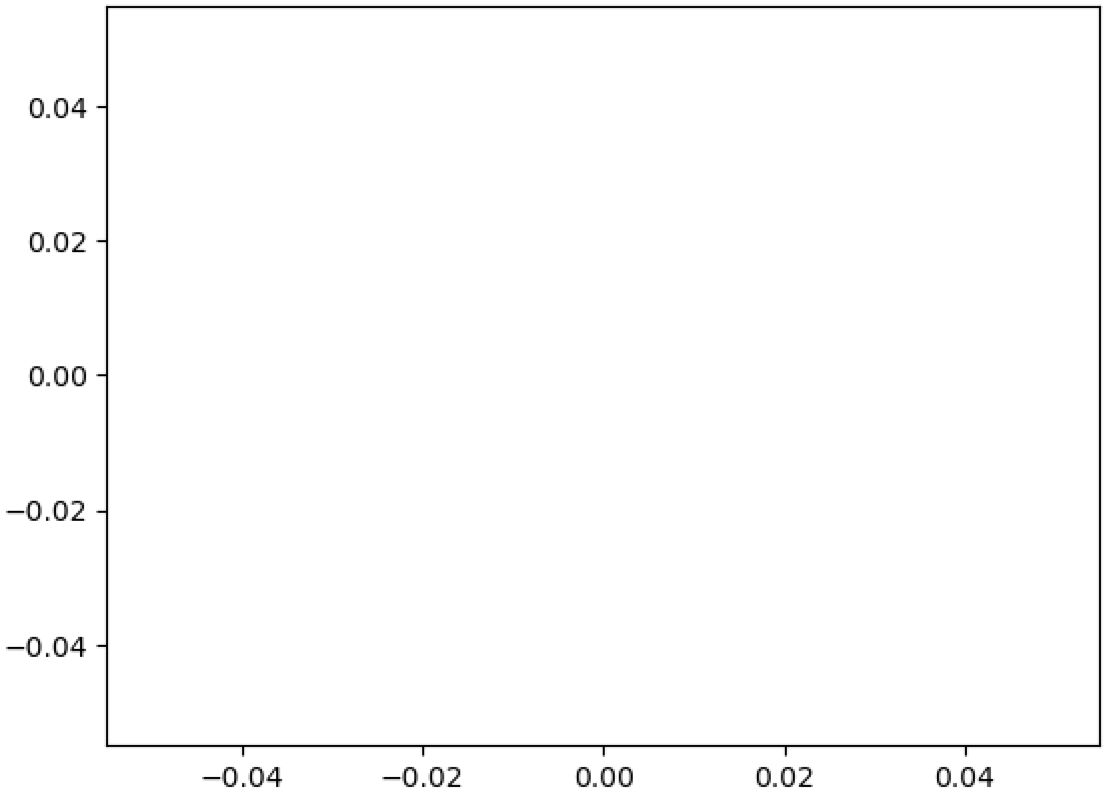


In [3]:

```
# to get rid of the bracket on top of the grpah
plt.plot();
```

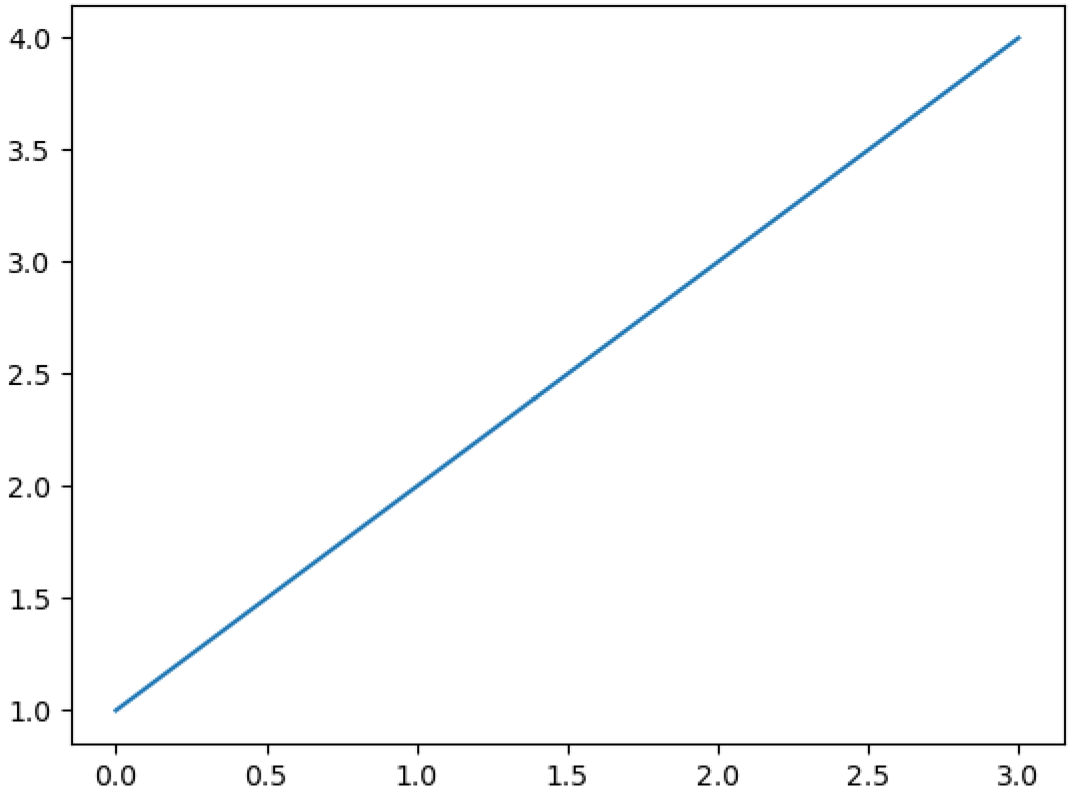


```
In [4]: plt.plot()
        plt.show()
```



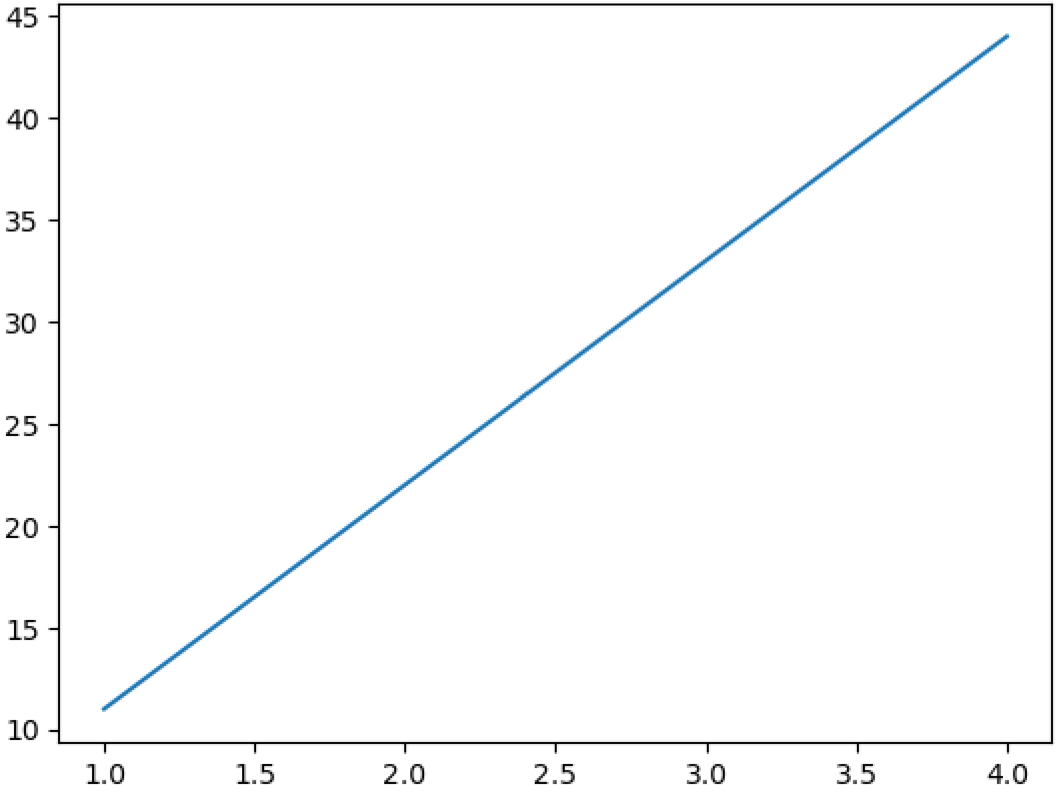
In [5]:

```
plt.plot([1, 2, 3, 4]);
```



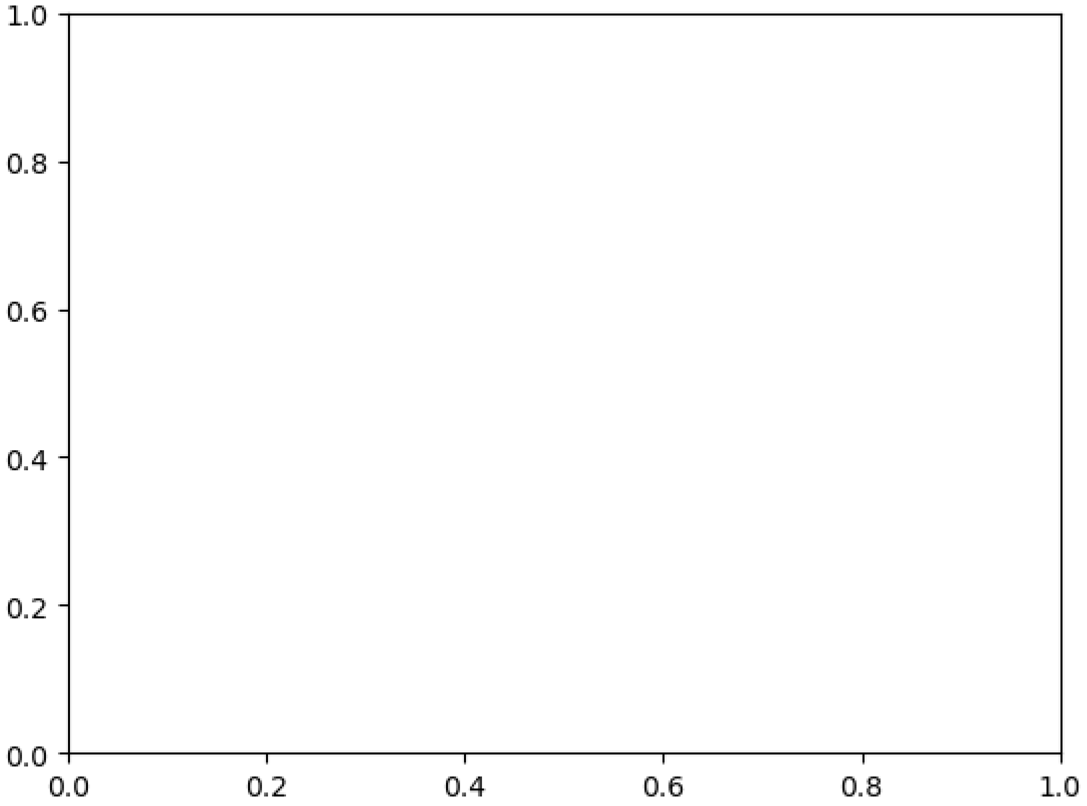
In [6]:

```python
x = [1, 2, 3, 4]
y = [11, 22, 33, 44]
plt.plot(x, y);
```



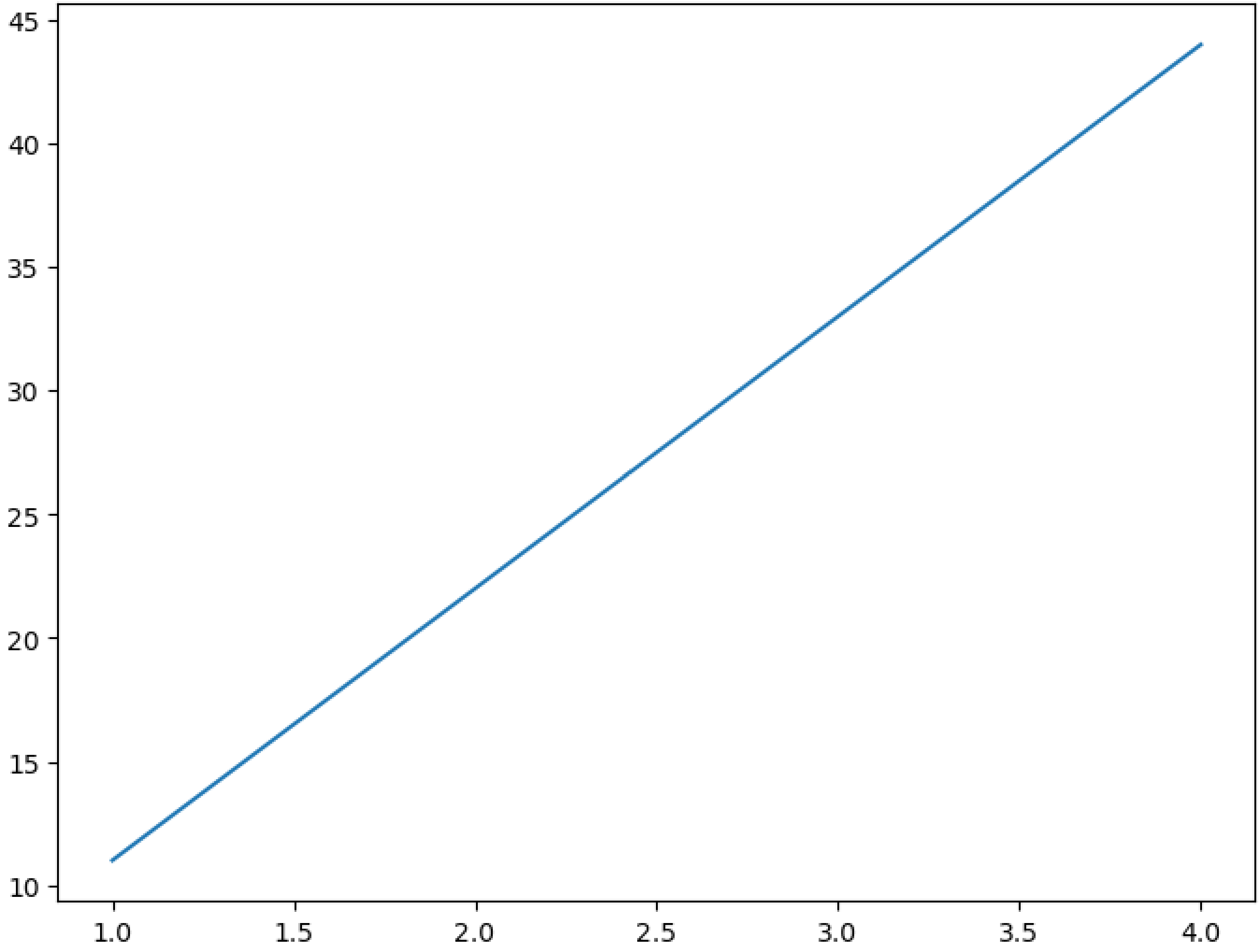
In [7]:

```
# 1st method
fig = plt.figure() # creates a figure
ax = fig.add_subplot() # adds some axes
plt.show()
```



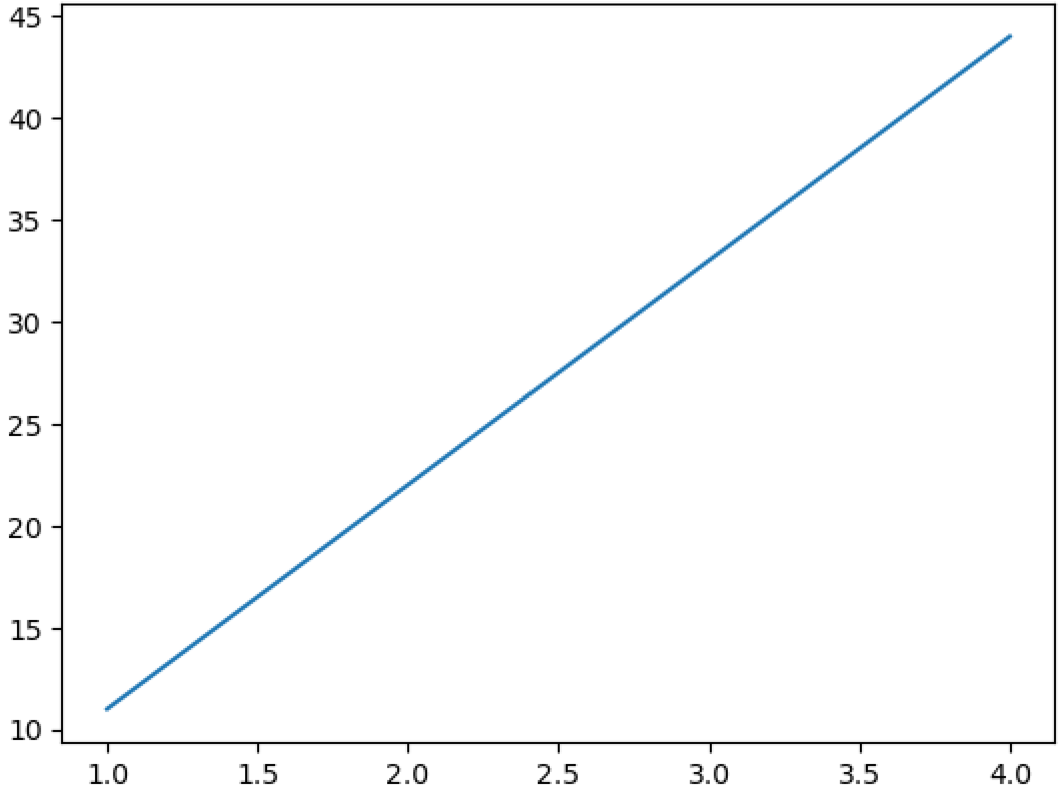
In [8]:

```
# 2nd method
fig = plt.figure() # creates a figure
ax = fig.add_axes([1, 1, 1, 1])
ax.plot(x, y) # add some data
plt.show()
```



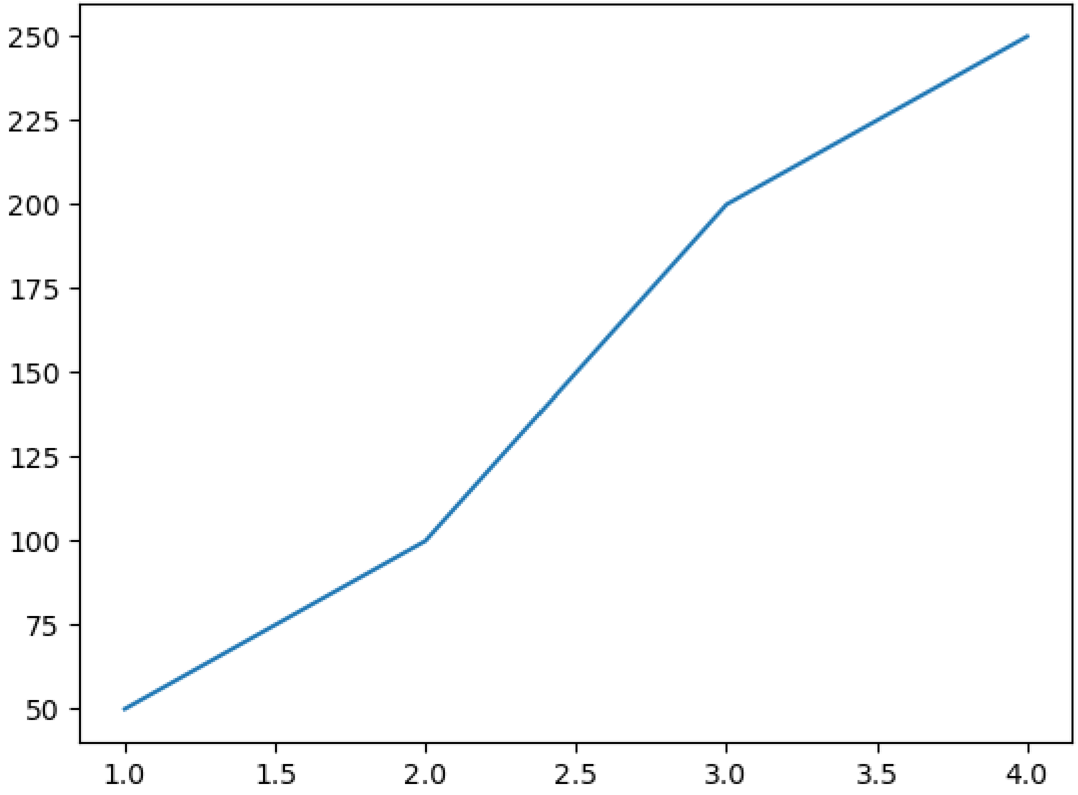
In [9]:

```
# 3rd method (recommended)
fig, ax = plt.subplots()
ax.plot(x, y); # add some data
```



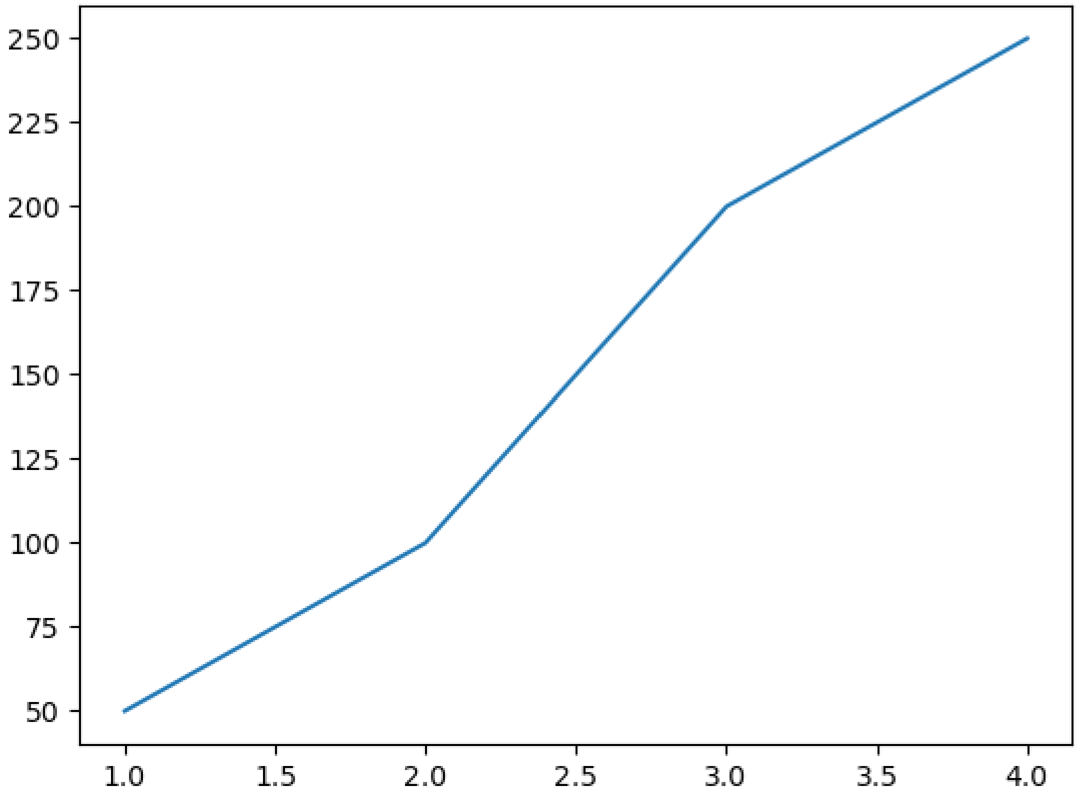
In [10]:

```
# 3rd method (recommended)
fig, ax = plt.subplots()
ax.plot(x, [50, 100, 200, 250]); # add some data
```



In [11]:
```
fig, ax = plt.subplots()
ax.plot(x, [50, 100, 200, 250]); # add some data
type(fig), type(ax)
```

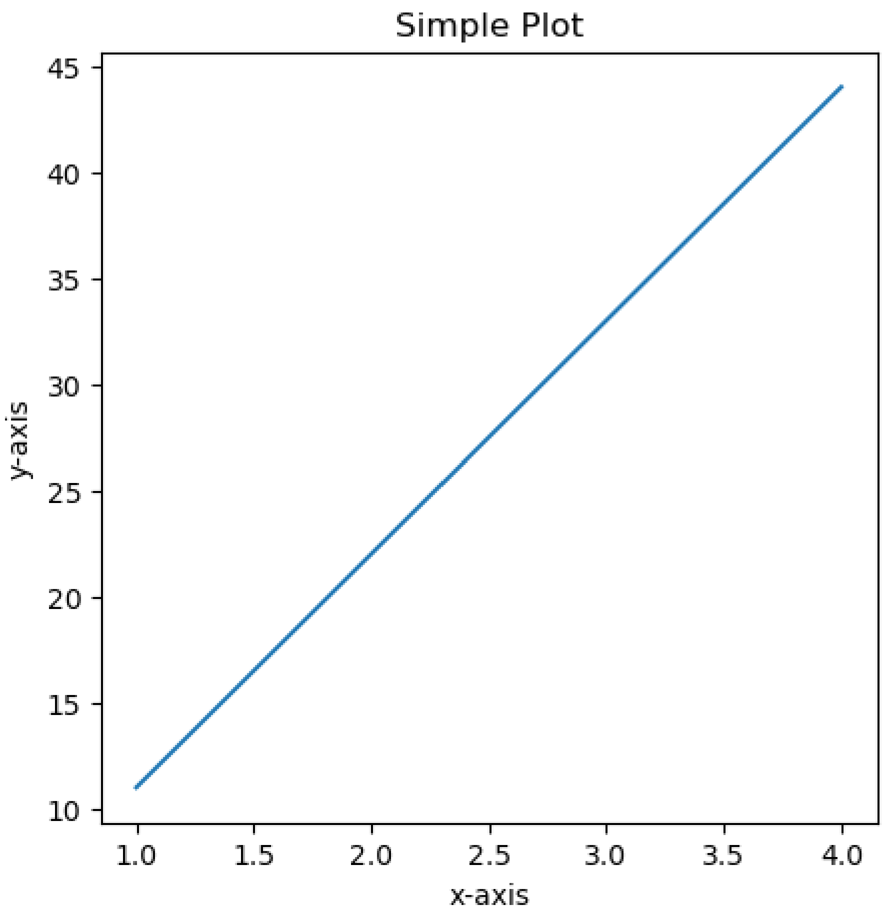
Out[11]:
```
(matplotlib.figure.Figure, matplotlib.axes._subplots.AxesSubplot)
```



Matplotlib example workflow

In [12]:

```python
#  0. import matplotlib and get it ready for plotting in Jupyter
%matplotlib inline
import matplotlib.pyplot as plt

# 1.  prepare data
x = [1, 2, 3, 4]
y = [11, 22, 33, 44]

# 2. setup plot
fig, ax = plt.subplots(figsize =(5, 5)) #width and height

# 3. plot data
ax.plot(x, y)

# 4. customize plot
ax.set(title='Simple Plot',
      xlabel='x-axis',
      ylabel='y-axis')


# 5. save and show (save the whole figure)
fig.savefig('images/sample_project1.png')
```



Making figures with Numpy arrays

we want: 1- line plot 2- scatter plot 3- bar plot 4- histogram 5-subplot

In [13]:
```
import numpy as np
```

In [14]:

```
# create some data
x = np.linspace(0, 10, 100)
x[: 10] # first 10
```

Out[14]:

```
array([0.        , 0.1010101 , 0.2020202 , 0.3030303 , 0.4040404 ,
       0.50505051, 0.60606061, 0.70707071, 0.80808081, 0.90909091])
```

In [15]:

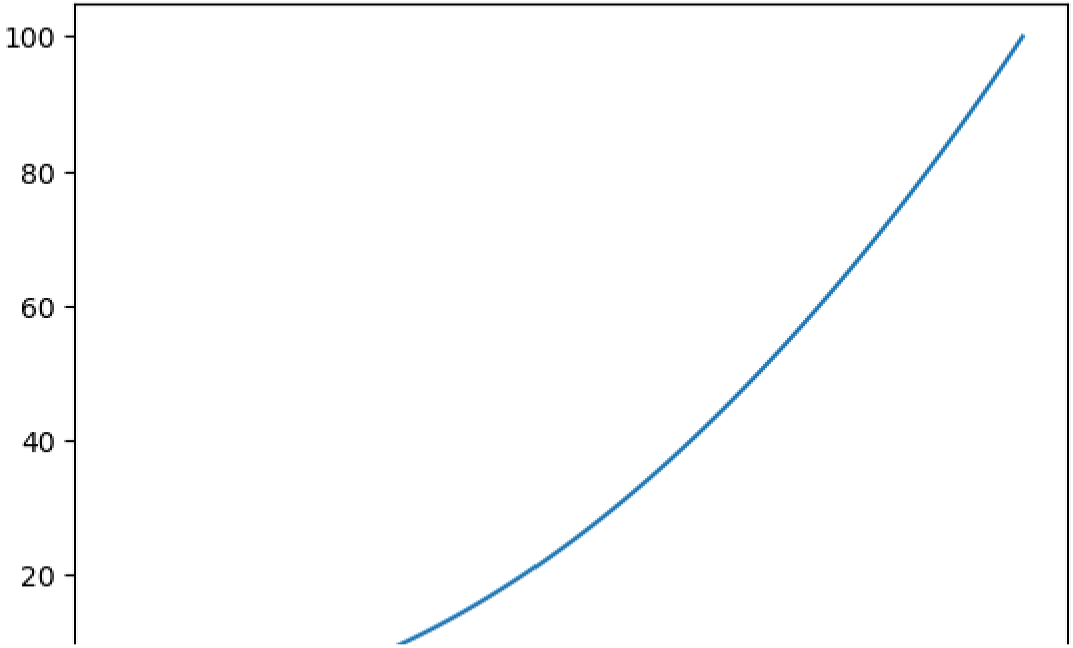
```
x = np.linspace(0, 10, 100)
x
```

Out[15]:

```
array([ 0.        ,  0.1010101 ,  0.2020202 ,  0.3030303 ,  0.4040404 ,
        0.50505051,  0.60606061,  0.70707071,  0.80808081,  0.90909091,
        1.01010101,  1.11111111,  1.21212121,  1.31313131,  1.41414141,
        1.51515152,  1.61616162,  1.71717172,  1.81818182,  1.91919192,
        2.02020202,  2.12121212,  2.22222222,  2.32323232,  2.42424242,
        2.52525253,  2.62626263,  2.72727273,  2.82828283,  2.92929293,
        3.03030303,  3.13131313,  3.23232323,  3.33333333,  3.43434343,
        3.53535354,  3.63636364,  3.73737374,  3.83838384,  3.93939394,
        4.04040404,  4.14141414,  4.24242424,  4.34343434,  4.44444444,
        4.54545455,  4.64646465,  4.74747475,  4.84848485,  4.94949495,
        5.05050505,  5.15151515,  5.25252525,  5.35353535,  5.45454545,
        5.55555556,  5.65656566,  5.75757576,  5.85858586,  5.95959596,
        6.06060606,  6.16161616,  6.26262626,  6.36363636,  6.46464646,
        6.56565657,  6.66666667,  6.76767677,  6.86868687,  6.96969697,
        7.07070707,  7.17171717,  7.27272727,  7.37373737,  7.47474747,
        7.57575758,  7.67676768,  7.77777778,  7.87878788,  7.97979798,
        8.08080808,  8.18181818,  8.28282828,  8.38383838,  8.48484848,
        8.58585859,  8.68686869,  8.78787879,  8.88888889,  8.98989899,
        9.09090909,  9.19191919,  9.29292929,  9.39393939,  9.49494949,
        9.5959596 ,  9.6969697 ,  9.7979798 ,  9.8989899 , 10.        ])
```

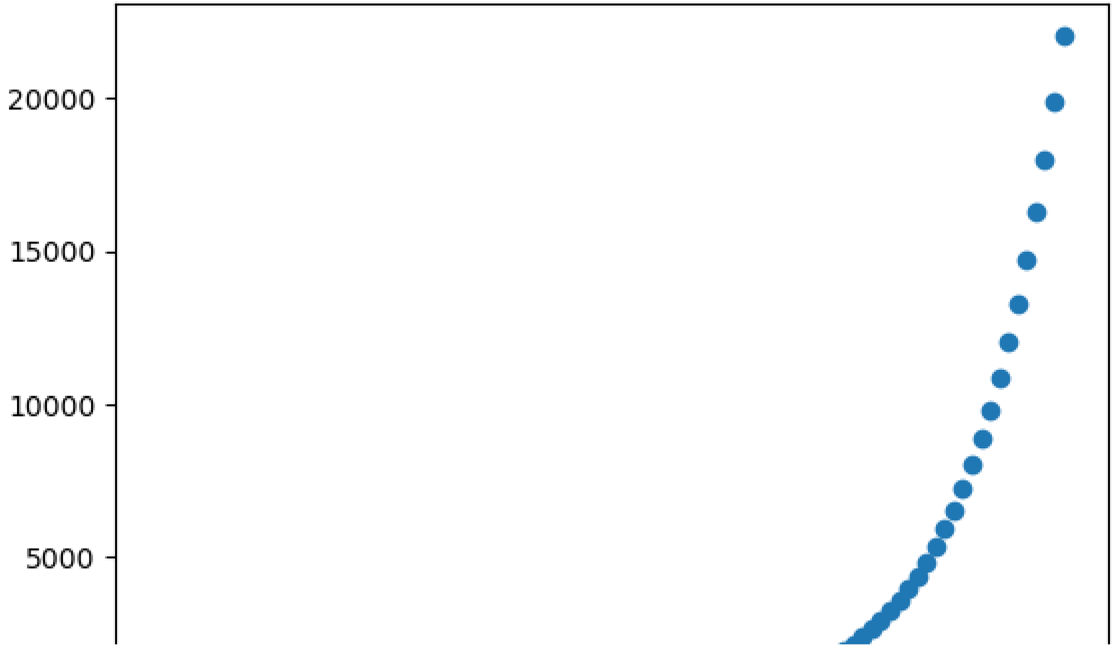
In [16]:

```
#  plot the data and create a line plot
fig, ax = plt.subplots()
ax.plot(x, x**2);

```



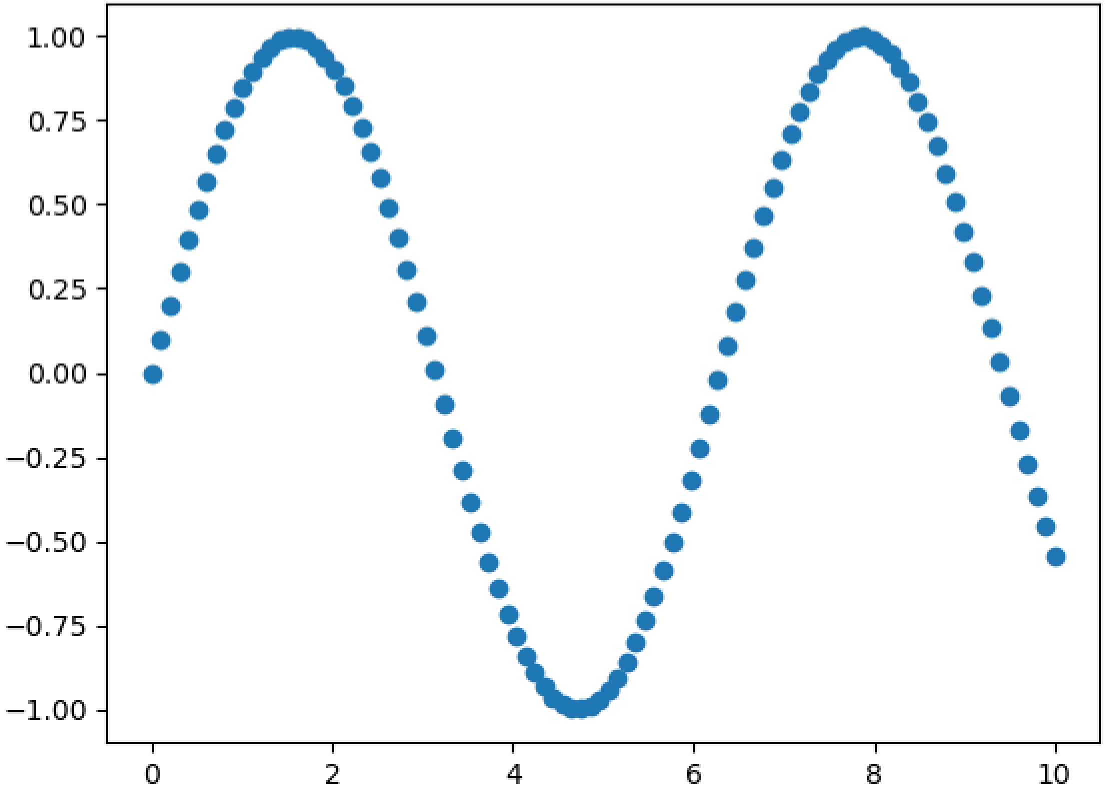
In [17]:

```
# use same data to make a scatter plot
fig, ax = plt.subplots()
ax.scatter(x, np.exp(x));
```



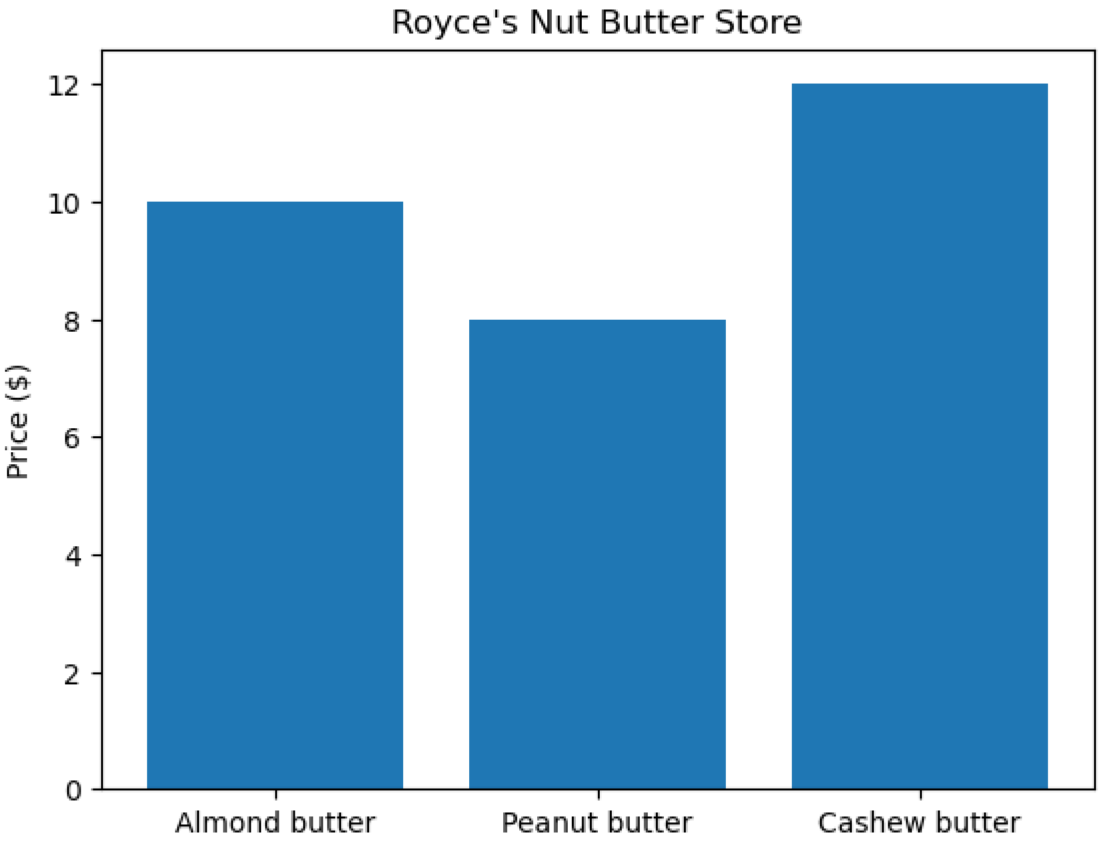
In [18]:

```
# another scatter plot
fig, ax = plt.subplots()
ax.scatter(x, np.sin(x));
```



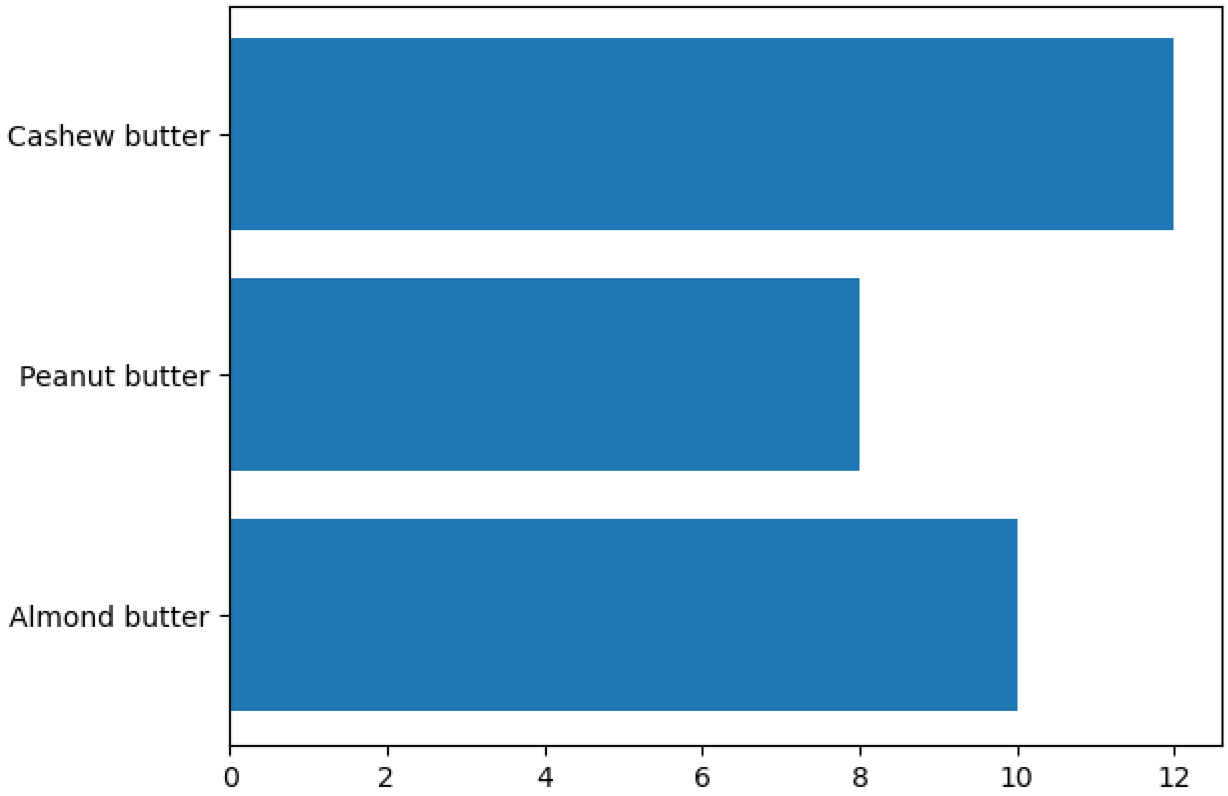
In [19]:

```
# make a plot from dictionary
nut_butter_prices = {'Almond butter': 10,
                    'Peanut butter': 8,
                    'Cashew butter': 12}
fig, ax = plt.subplots()
ax.bar(nut_butter_prices.keys(),  nut_butter_prices.values())
ax.set(title = "Royce's Nut Butter Store",
      ylabel = 'Price ($)'
    );

```



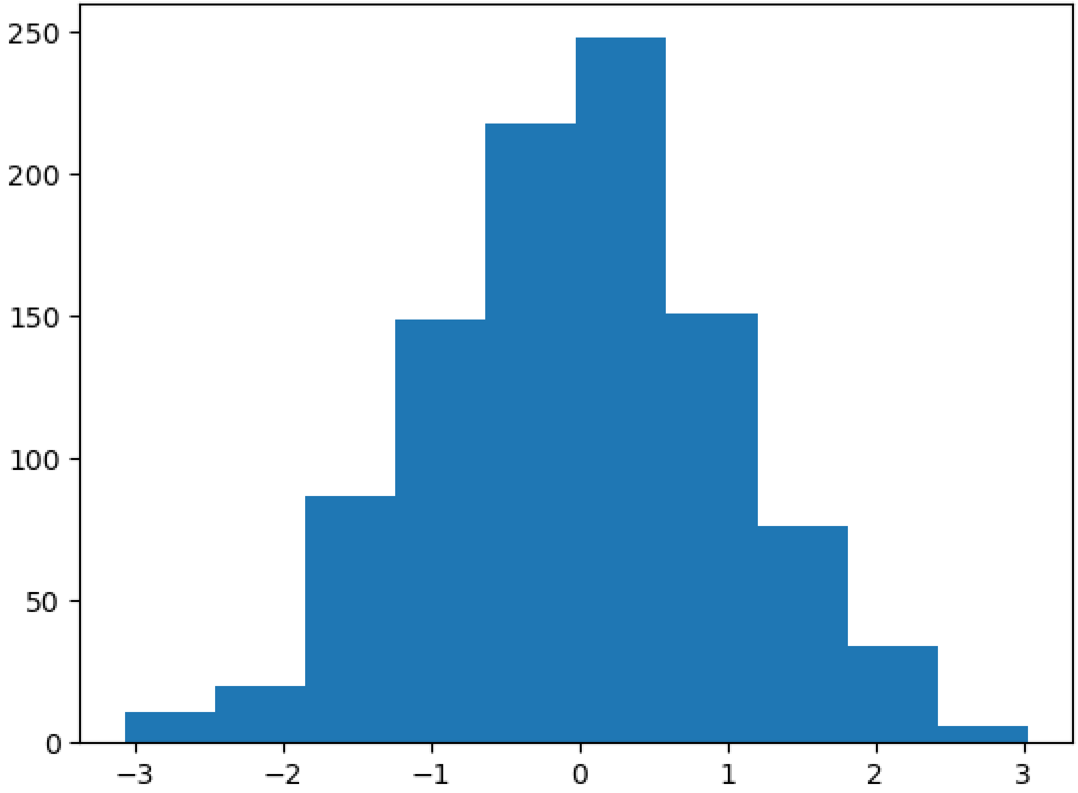
In [20]:

```
fig, ax = plt.subplots()
ax.barh(list(nut_butter_prices.keys()), list(nut_butter_prices.values()));
```



In [21]:

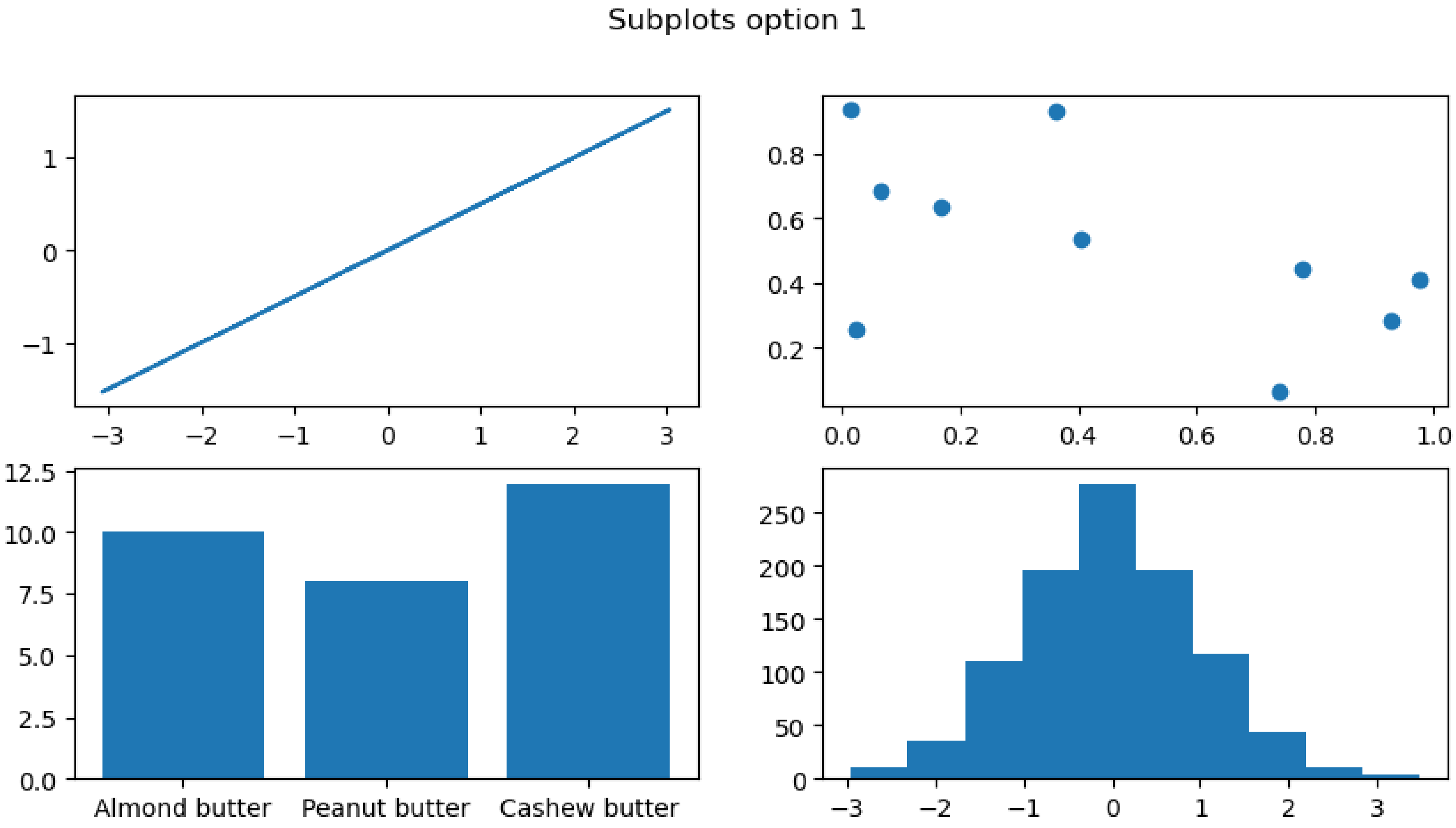
```
# make saome data for histograms and plot it
x = np.random.randn(1000)
fig, ax = plt.subplots()
ax.hist(x);
```



Two options for subplots

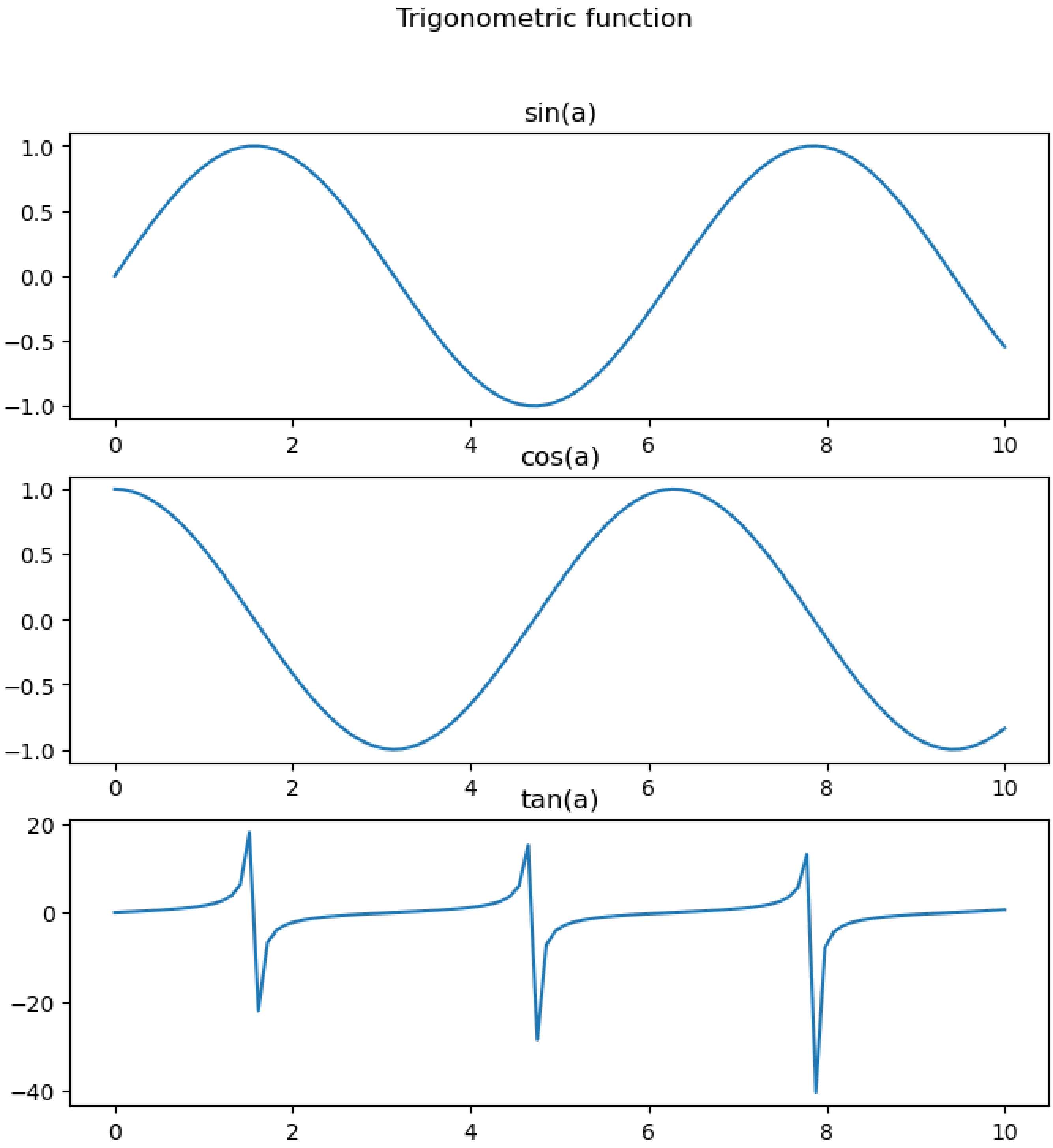
In [22]:

```
# subplot option 1
fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(nrows = 2,
                                             ncols = 2,
                                             figsize = (10, 5))
# plot to each different axis
ax1.plot(x, x/2);
ax2.scatter(np.random.random(10), np.random.random(10));
ax3.bar(nut_butter_prices.keys(), nut_butter_prices.values());
ax4.hist(np.random.randn(1000));
fig.suptitle('Subplots option 1');
```



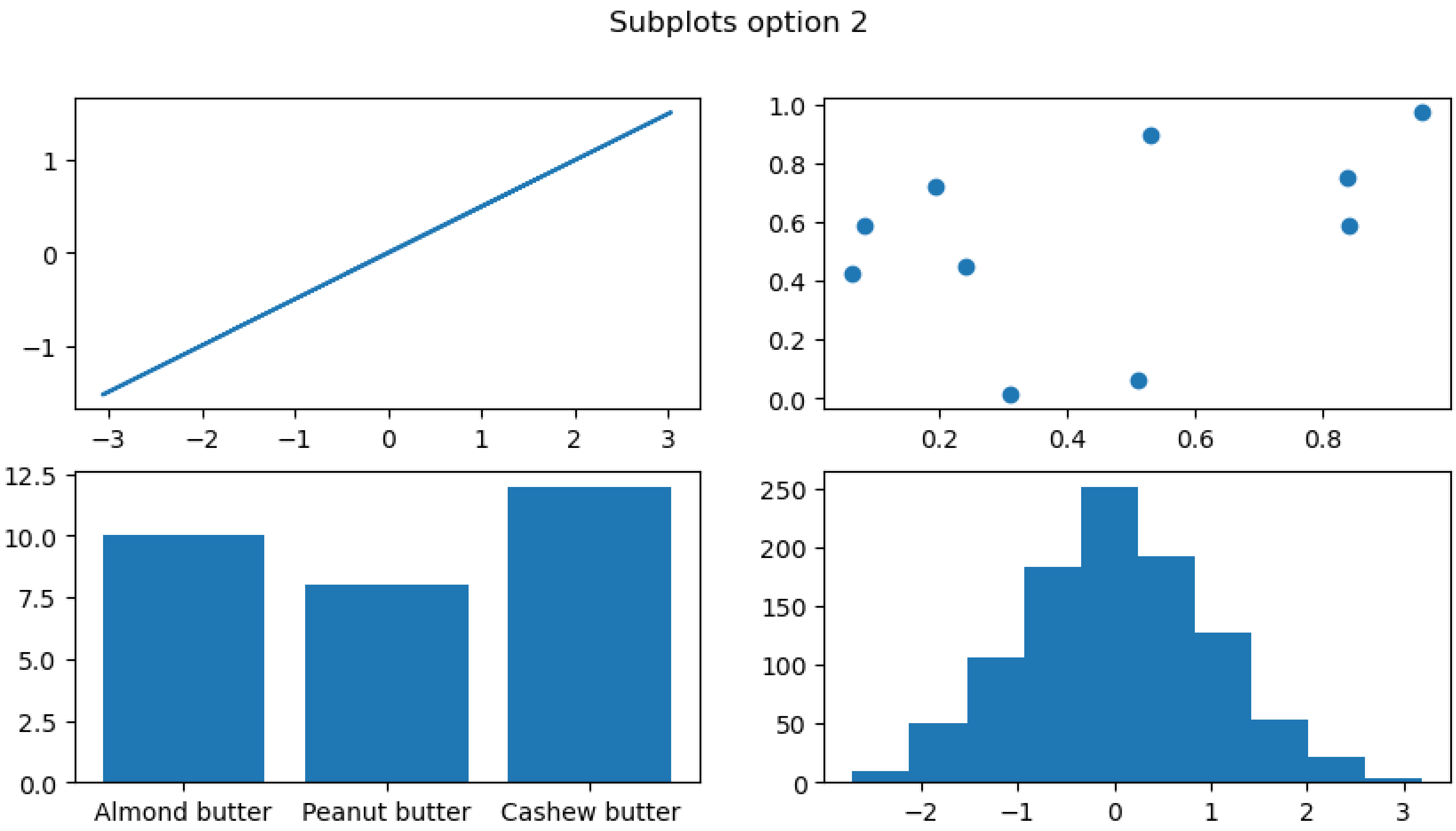
In [23]:

```
a = np.linspace(0, 10, 100)
y1 = np.sin(a)
y2 = np.cos(a)
y3 = np.tan(a)

fig, ax = plt.subplots(3, 1, figsize = (8, 8))
ax[0].plot(a, y1)
ax[1].plot(a, y2)
ax[2].plot(a, y3)

ax[0].set_title('sin(a)')
ax[1].set_title('cos(a)')
ax[2].set_title('tan(a)')
fig.suptitle('Trigonometric function')

plt.show()

```


Trigonometric function
sin(a)
1.0
0.5
0.0
−0.5
−1.0
0
2
4
6
8
10
cos(a)
1.0
0.5
0.0
−0.5
−1.0
0
2
4
6
8
10
tan(a)
20
0
−20
−40
0
2
4
6
8
10

In [24]:

```
# subplot option 2
fig, ax = plt.subplots(nrows = 2,
                       ncols = 2,
                       figsize = (10, 5))
# plot to each different index
ax[0, 0].plot(x, x/2);
ax[0, 1].scatter(np.random.random(10), np.random.random(10));
ax[1, 0].bar(nut_butter_prices.keys(), nut_butter_prices.values());
ax[1, 1].hist(np.random.randn(1000));
fig.suptitle('Subplots option 2');
```



Plotting from pandas Dataframe

In [25]:

```
import pandas as pd
```

In [26]:

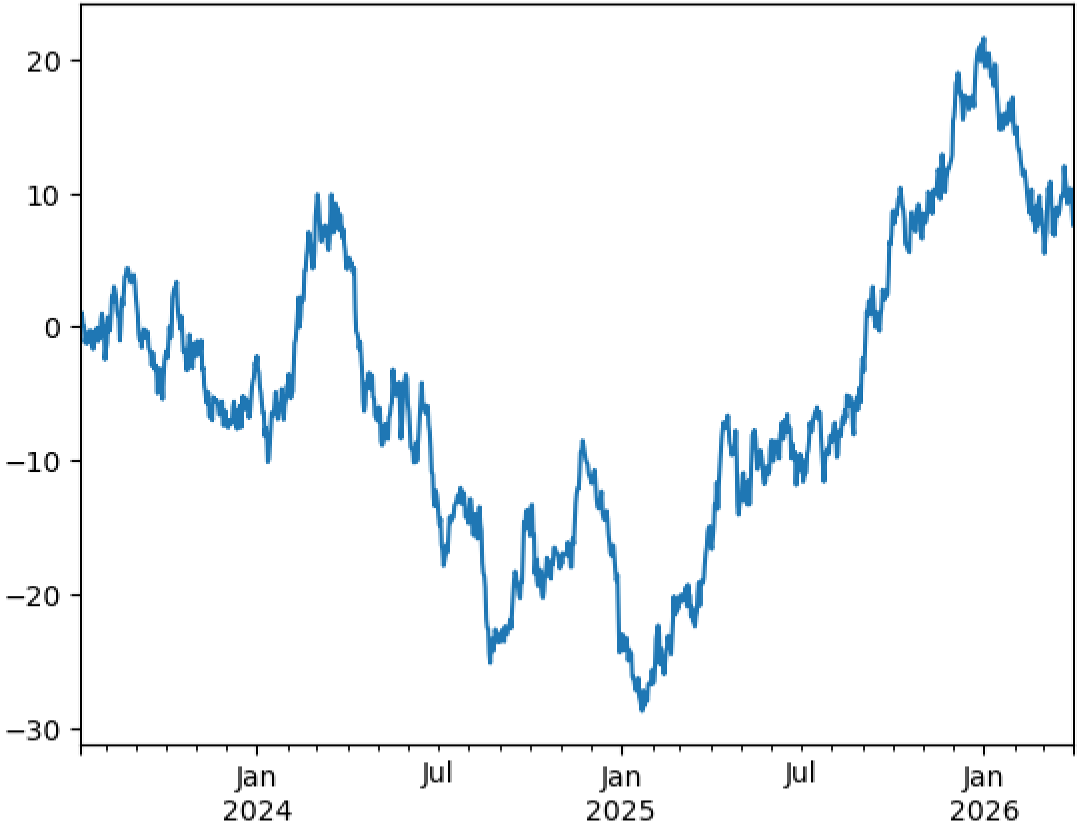
```
# make a dataframe
car_sales = pd.read_csv('car-sales.csv')
car_sales
```

Out[26]:

	Make	Colour	Odometer (KM)	Doors	Price
0	Toyota	White	150043	4	$4,000.00
1	Honda	Red	87899	4	$5,000.00
2	Toyota	Blue	32549	3	$7,000.00
3	BMW	Black	11179	5	$22,000.00
4	Nissan	White	213095	4	$3,500.00
5	Toyota	Green	99213	4	$4,500.00
6	Honda	Blue	45698	4	$7,500.00
7	Honda	Blue	54738	4	$7,000.00
8	Toyota	White	60000	4	$6,250.00
9	Nissan	White	31600	4	$9,700.00

In [27]:

```
ts = pd.Series(np.random.randn(1000),
               index = pd.date_range('7/7/2023', periods = 1000))
ts = ts.cumsum()
ts.plot();
```



In [28]:

```
car_sales
```

Out[28]:

	Make	Colour	Odometer (KM)	Doors	Price
0	Toyota	White	150043	4	$4,000.00
1	Honda	Red	87899	4	$5,000.00
2	Toyota	Blue	32549	3	$7,000.00
3	BMW	Black	11179	5	$22,000.00
4	Nissan	White	213095	4	$3,500.00
5	Toyota	Green	99213	4	$4,500.00
6	Honda	Blue	45698	4	$7,500.00
7	Honda	Blue	54738	4	$7,000.00
8	Toyota	White	60000	4	$6,250.00
9	Nissan	White	31600	4	$9,700.00

In [29]:

```
car_sales['Price'] = car_sales['Price'].str.replace('[\$\,\.]', '')
car_sales
```

```
C:\Users\USER\AppData\Local\Temp\ipykernel_6116\1919509590.py:1: FutureWarning: The default value of reg
ex will change from True to False in a future version.
  car_sales['Price'] = car_sales['Price'].str.replace('[\$\,\.]', '')
```

Out[29]:

	Make	Colour	Odometer (KM)	Doors	Price
0	Toyota	White	150043	4	400000
1	Honda	Red	87899	4	500000
2	Toyota	Blue	32549	3	700000
3	BMW	Black	11179	5	2200000
4	Nissan	White	213095	4	350000
5	Toyota	Green	99213	4	450000
6	Honda	Blue	45698	4	750000
7	Honda	Blue	54738	4	700000
8	Toyota	White	60000	4	625000
9	Nissan	White	31600	4	970000

In [30]:

```
type(car_sales['Price'][0])
```

Out[30]:

```
str
```

In [31]:

```python
# Remove last two zeros
car_sales['Price'] = car_sales['Price'].str[:-3]
car_sales
```

Out[31]:

	Make	Colour	Odometer (KM)	Doors	Price
0	Toyota	White	150043	4	4000
1	Honda	Red	87899	4	5000
2	Toyota	Blue	32549	3	7000
3	BMW	Black	11179	5	22000
4	Nissan	White	213095	4	3500
5	Toyota	Green	99213	4	4500
6	Honda	Blue	45698	4	7500
7	Honda	Blue	54738	4	7000
8	Toyota	White	60000	4	6250
9	Nissan	White	31600	4	9700

In [32]:

```
car_sales['Sale Date'] = pd.date_range('3/7/2023', periods = len(car_sales))
car_sales
```

Out[32]:

	Make	Colour	Odometer (KM)	Doors	Price	Sale Date
0	Toyota	White	150043	4	4000	2023-03-07
1	Honda	Red	87899	4	5000	2023-03-08
2	Toyota	Blue	32549	3	7000	2023-03-09
3	BMW	Black	11179	5	22000	2023-03-10
4	Nissan	White	213095	4	3500	2023-03-11
5	Toyota	Green	99213	4	4500	2023-03-12
6	Honda	Blue	45698	4	7500	2023-03-13
7	Honda	Blue	54738	4	7000	2023-03-14
8	Toyota	White	60000	4	6250	2023-03-15
9	Nissan	White	31600	4	9700	2023-03-16

In [33]:

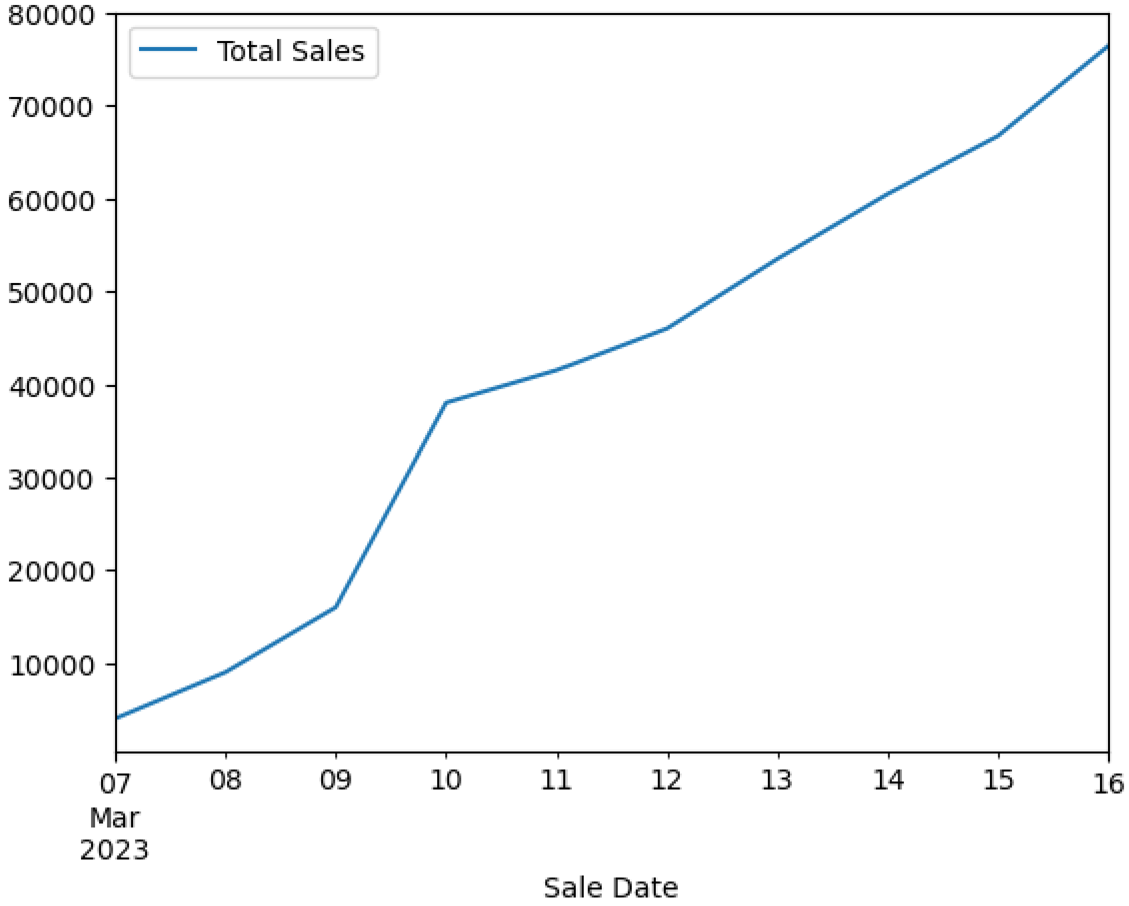
```
car_sales['Total Sales'] = car_sales['Price'].astype(int).cumsum()
car_sales
```

Out[33]:

	Make	Colour	Odometer (KM)	Doors	Price	Sale Date	Total Sales
0	Toyota	White	150043	4	4000	2023-03-07	4000
1	Honda	Red	87899	4	5000	2023-03-08	9000
2	Toyota	Blue	32549	3	7000	2023-03-09	16000
3	BMW	Black	11179	5	22000	2023-03-10	38000
4	Nissan	White	213095	4	3500	2023-03-11	41500
5	Toyota	Green	99213	4	4500	2023-03-12	46000
6	Honda	Blue	45698	4	7500	2023-03-13	53500
7	Honda	Blue	54738	4	7000	2023-03-14	60500
8	Toyota	White	60000	4	6250	2023-03-15	66750
9	Nissan	White	31600	4	9700	2023-03-16	76450

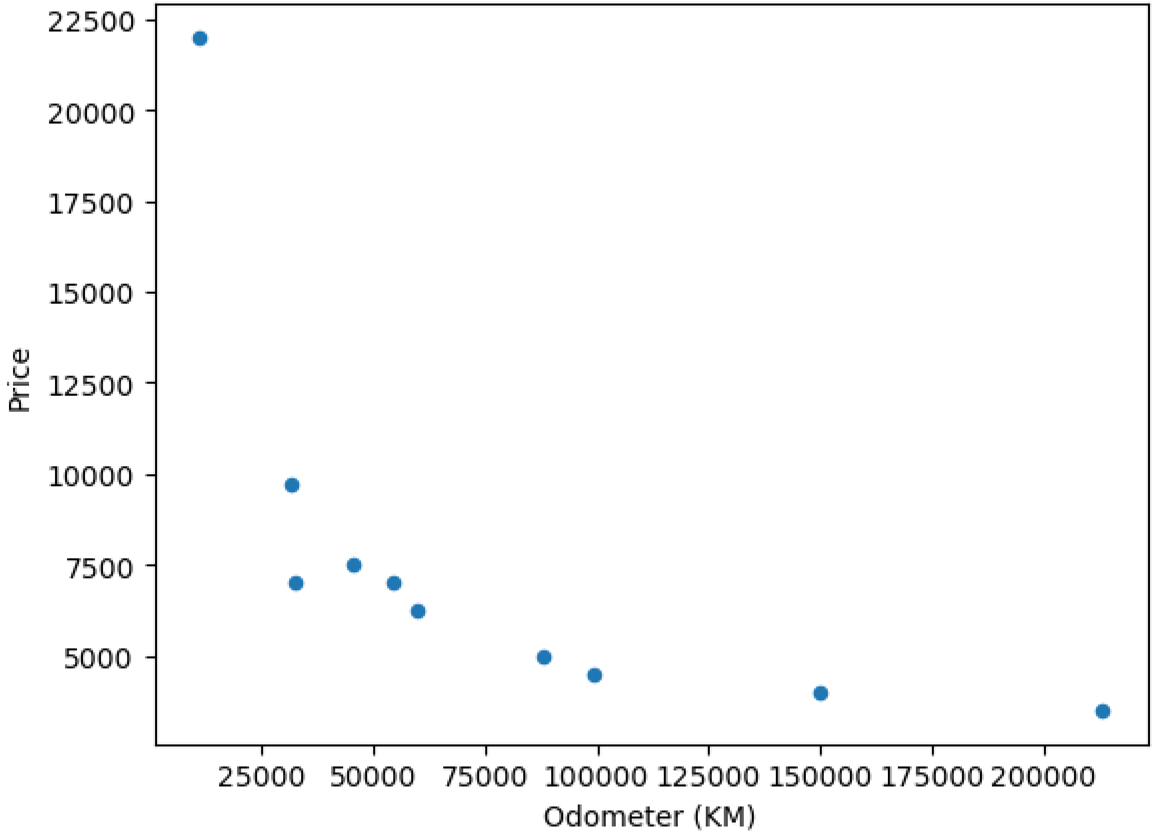
In [34]:

```
# Let's plot the total sales
car_sales.plot(x = 'Sale Date', y = 'Total Sales' );
```



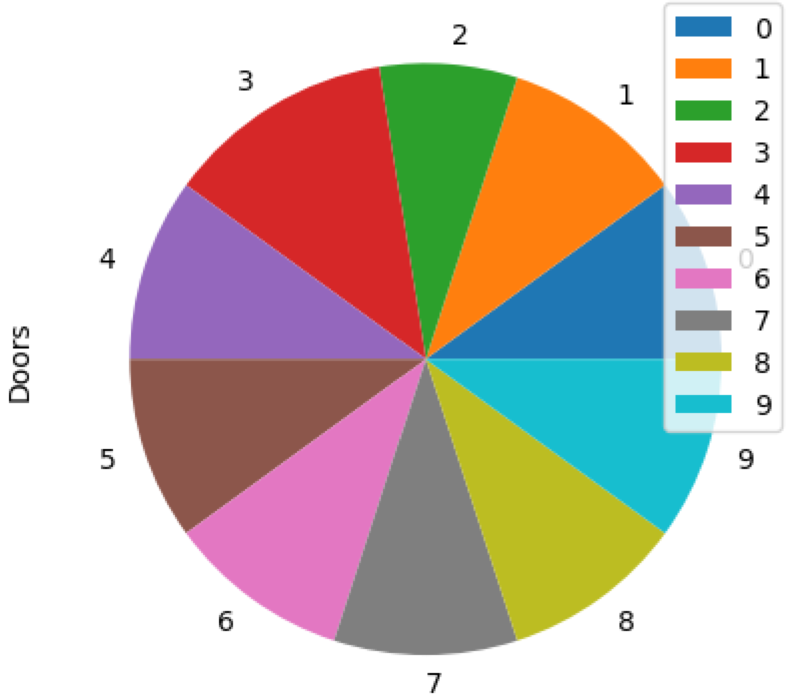
In [35]:

```
# reasssign price column to int
car_sales['Price'] = car_sales['Price'].astype(int)
# plot scatter plot with price column as numeric
car_sales.plot(x = 'Odometer (KM)', y = 'Price', kind = 'scatter' );

```



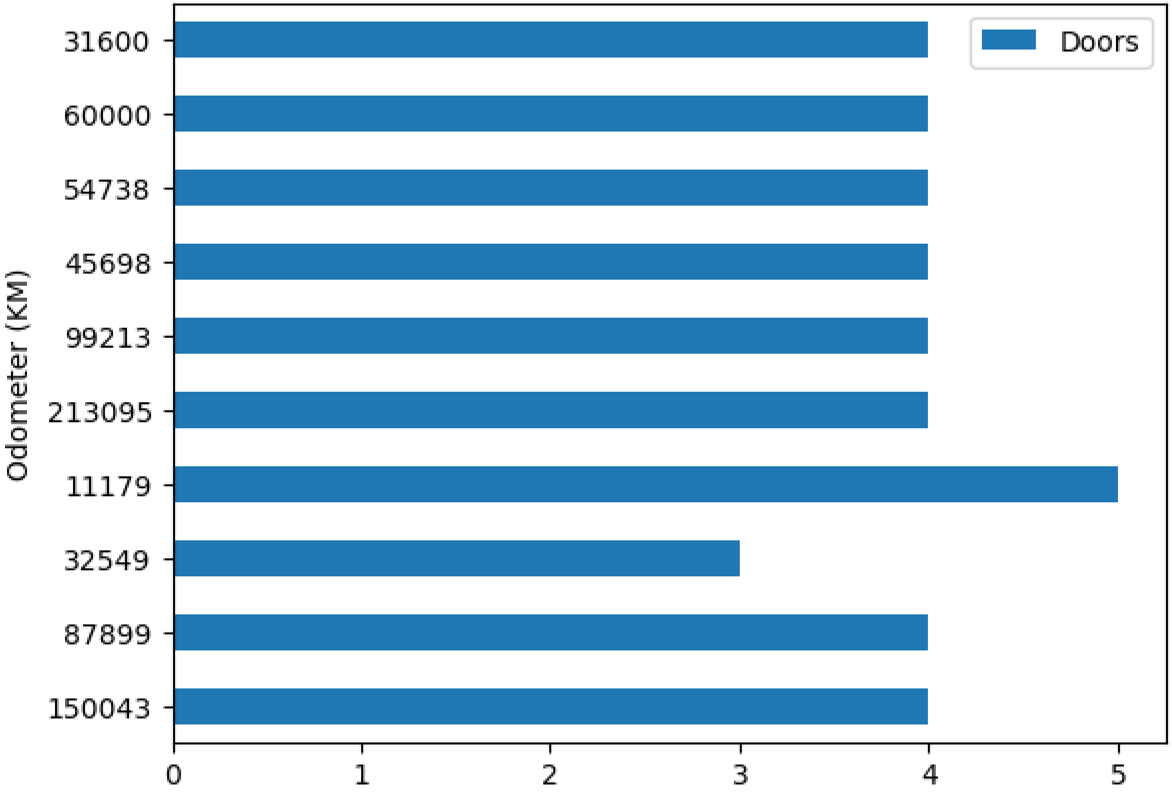
In [36]:

```python
car_sales.plot(x = 'Odometer (KM)', y = 'Doors', kind = 'pie');
```



In [37]:

```
car_sales.plot.barh(x = 'Odometer (KM)', y = 'Doors');
```



In [38]:

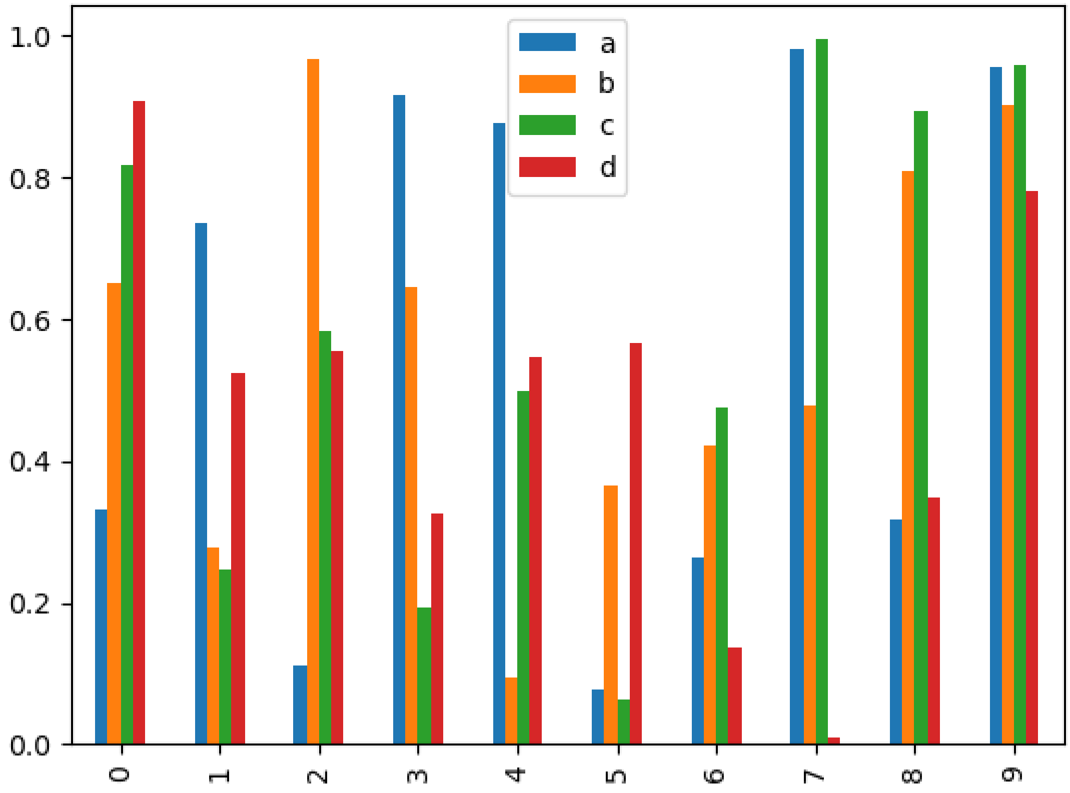
```python
# how about a bar graph?
x = np.random.rand(10, 4)
x

# turn it into dataframe
df = pd.DataFrame(x, columns = ['a', 'b', 'c', 'd'])
df
```

Out[38]:

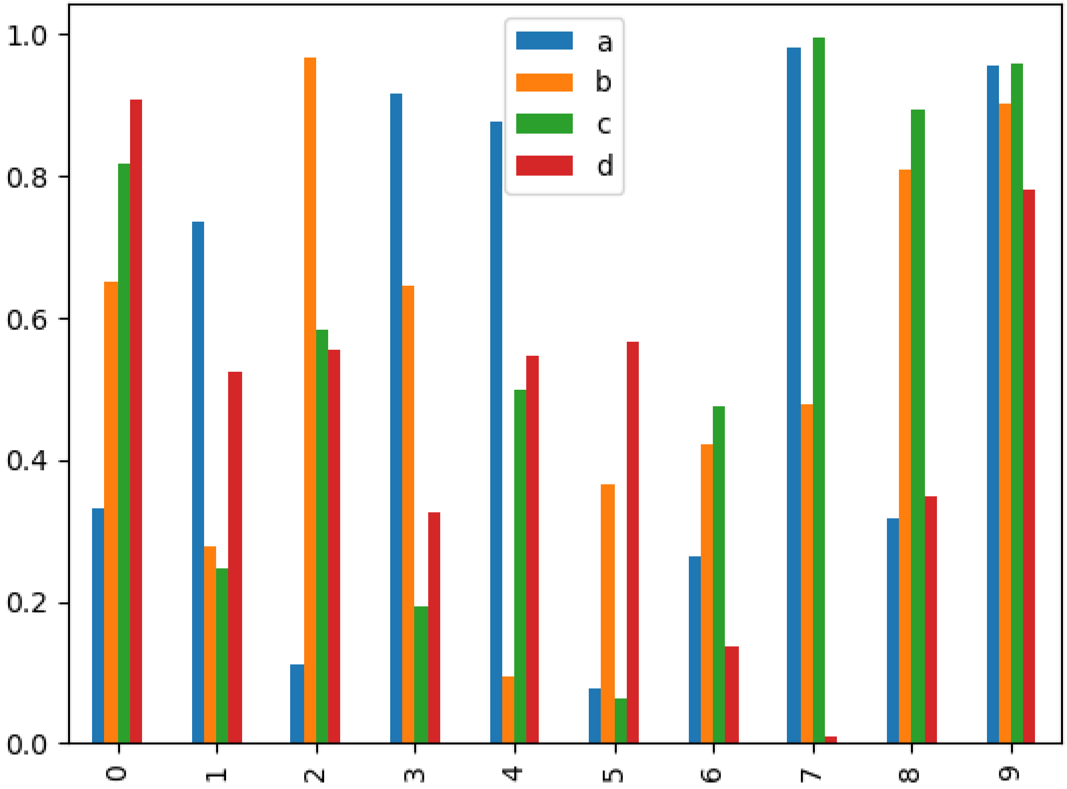
	a	b	c	d
0	0.332639	0.650146	0.816496	0.906705
1	0.734118	0.277576	0.246242	0.522385
2	0.111708	0.965688	0.582303	0.555258
3	0.915503	0.645103	0.192248	0.326236
4	0.876303	0.093955	0.498158	0.544852
5	0.078063	0.364316	0.061881	0.564708
6	0.263540	0.420879	0.475377	0.135933
7	0.981092	0.477656	0.994186	0.009853
8	0.316902	0.808801	0.893701	0.347658
9	0.954552	0.902423	0.958991	0.781712

In [39]:

```
df.plot.bar();
```



In [40]:
```
df.plot(kind = 'bar');
```



In [41]:

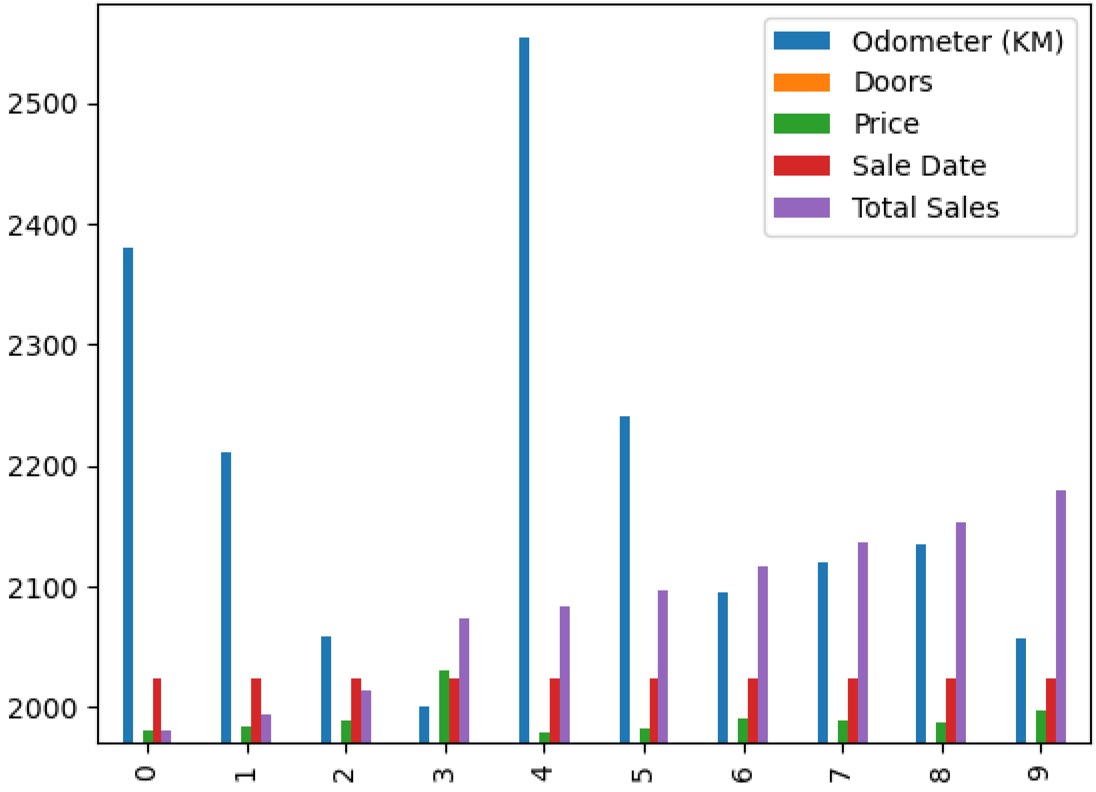
```
car_sales
```

Out[41]:

	Make	Colour	Odometer (KM)	Doors	Price	Sale Date	Total Sales
0	Toyota	White	150043	4	4000	2023-03-07	4000
1	Honda	Red	87899	4	5000	2023-03-08	9000
2	Toyota	Blue	32549	3	7000	2023-03-09	16000
3	BMW	Black	11179	5	22000	2023-03-10	38000
4	Nissan	White	213095	4	3500	2023-03-11	41500
5	Toyota	Green	99213	4	4500	2023-03-12	46000
6	Honda	Blue	45698	4	7500	2023-03-13	53500
7	Honda	Blue	54738	4	7000	2023-03-14	60500
8	Toyota	White	60000	4	6250	2023-03-15	66750
9	Nissan	White	31600	4	9700	2023-03-16	76450

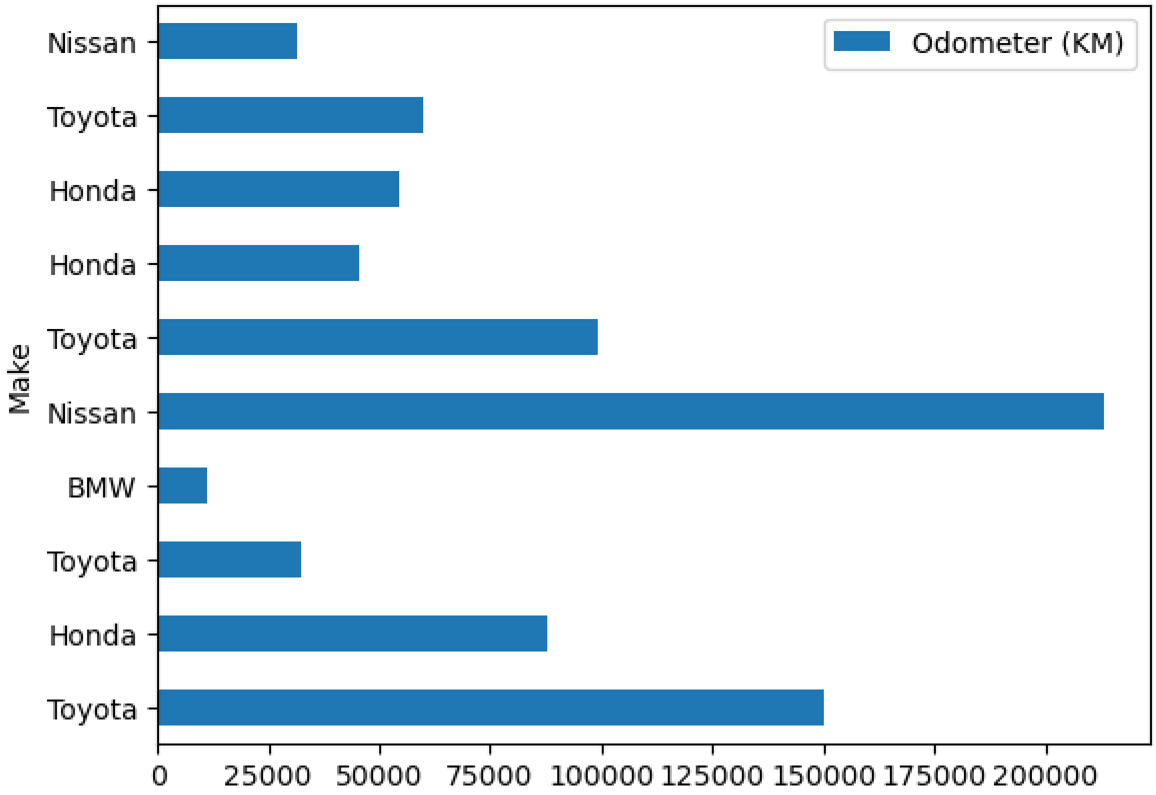
In [42]:

```
car_sales.plot.bar();
```



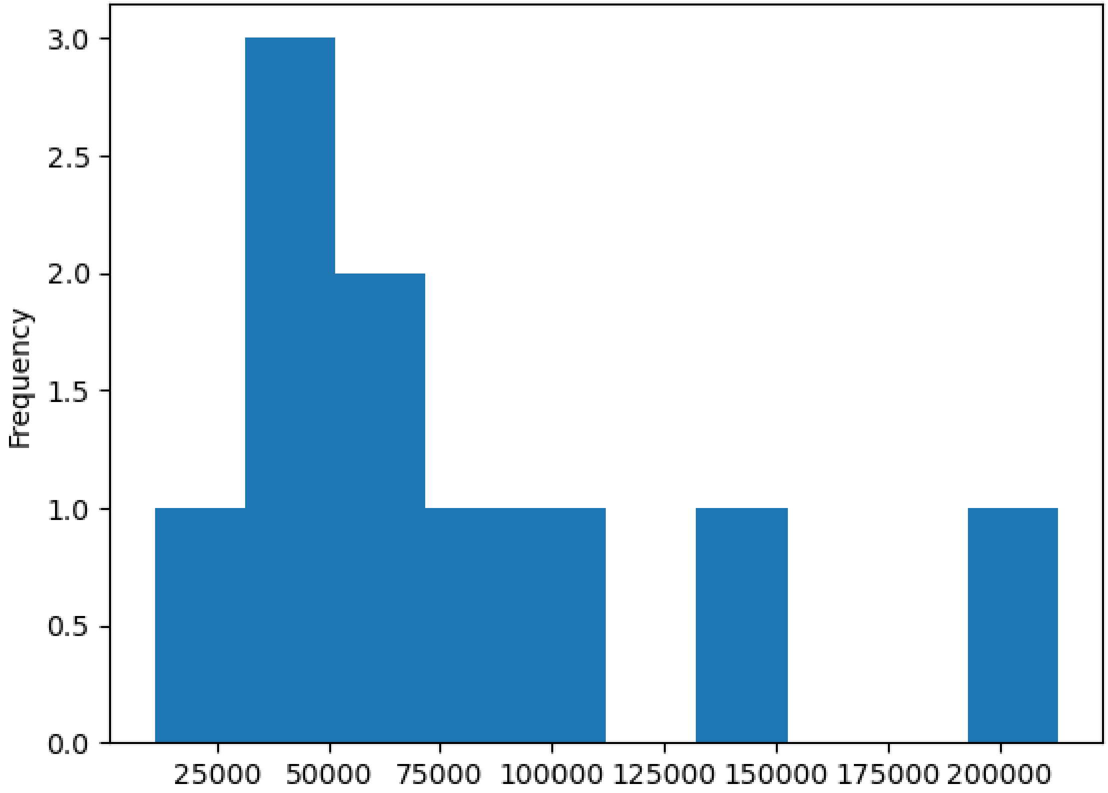
In [43]:

```
car_sales.plot(x ='Make', y = 'Odometer (KM)', kind ='barh');
```



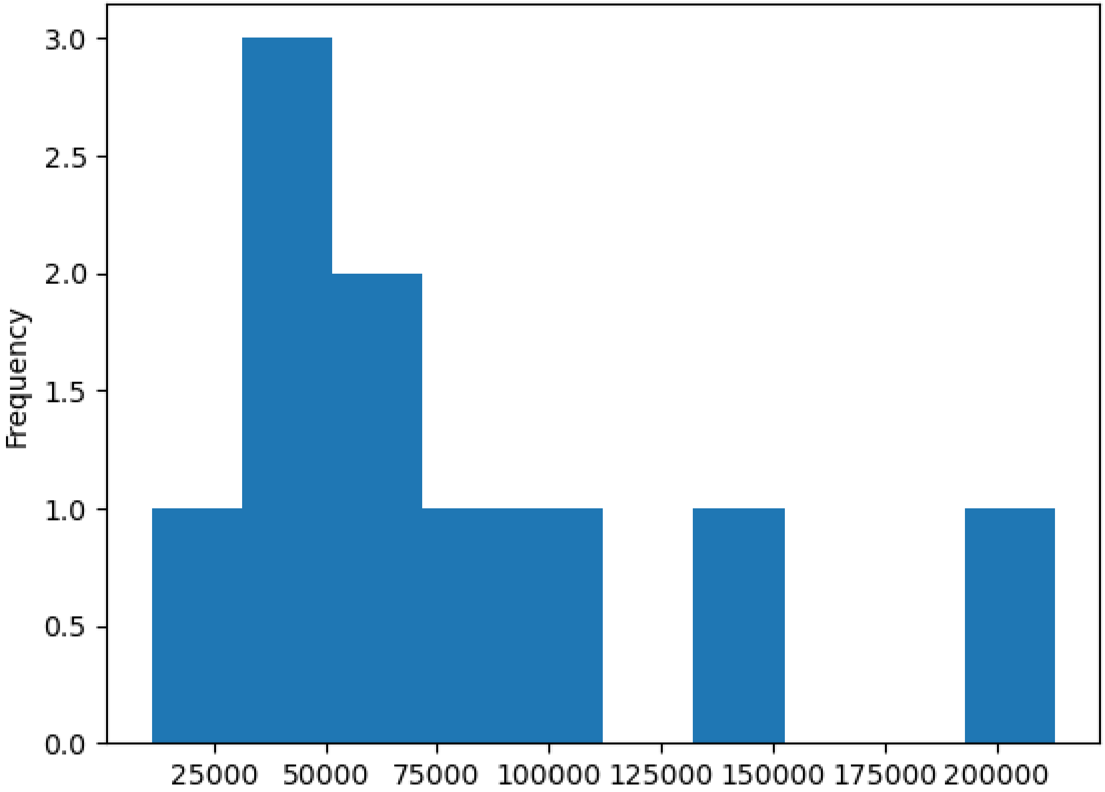
In [44]:

```
# how about histograms
car_sales['Odometer (KM)'].plot.hist();
```



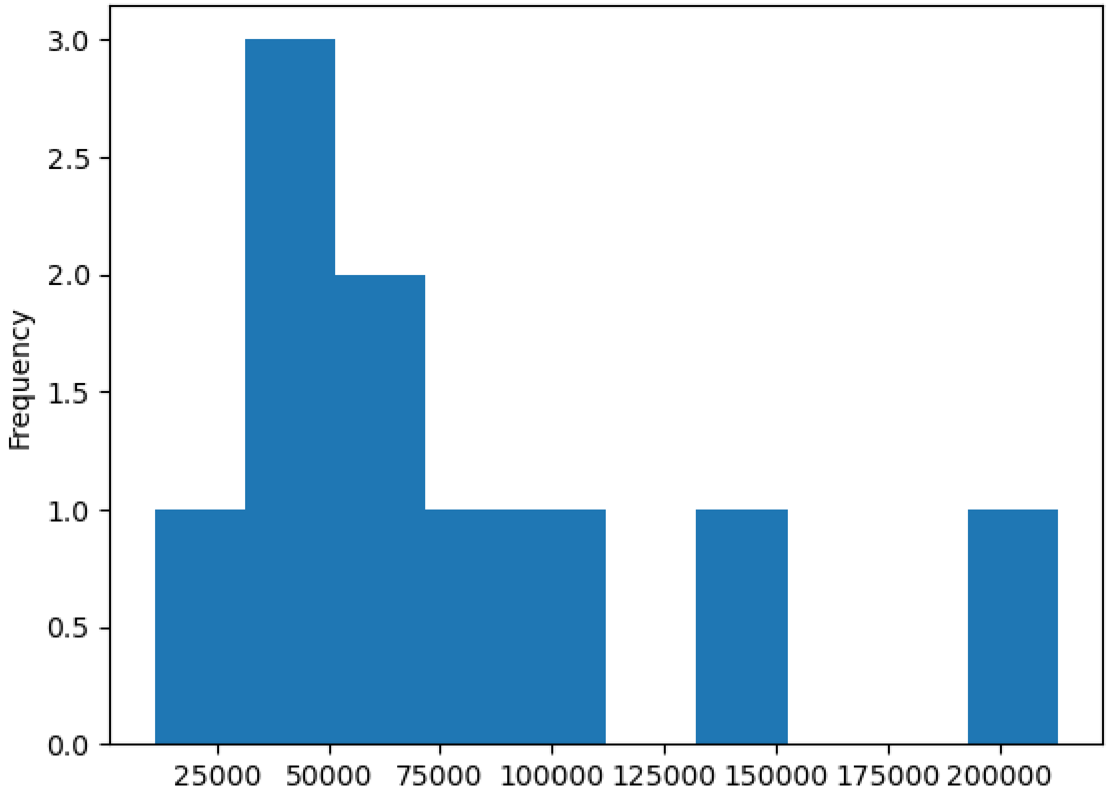
In [45]:

```
car_sales['Odometer (KM)'].plot(kind ='hist');
```



In [46]:

```python
car_sales['Odometer (KM)'].plot(bins = 10, kind ='hist');
```



In [47]:

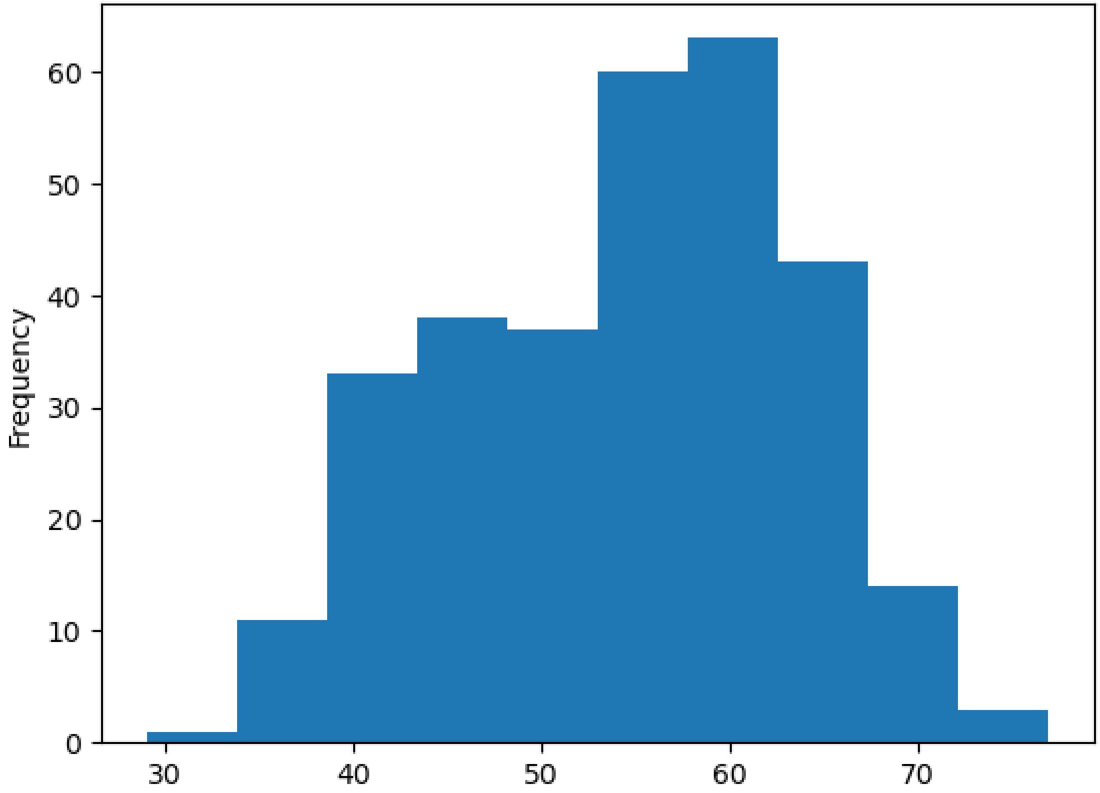
```python
# let's try on another dataset
heart_disease = pd.read_csv('heart-disease.csv')
heart_disease.head()
```

Out[47]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

In [48]:

```python
# create a histogram
heart_disease['age'].plot.hist();
```



In [49]:

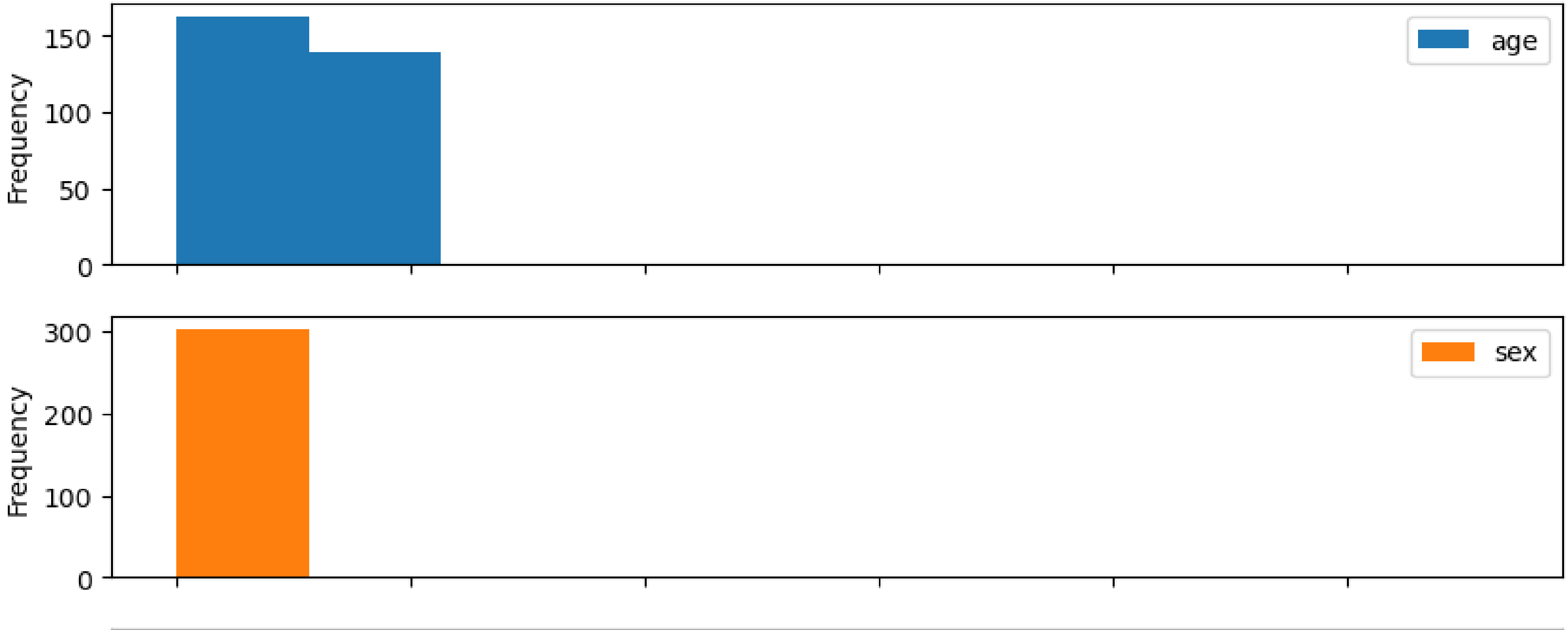
```python
heart_disease.head()
```

Out[49]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

In [50]:

```
heart_disease.plot.hist(figsize = (10, 30), subplots = True);
```



which one should you use? (pyplot vs matplotlib OO method?)

1. when plotting something quickly, okay to use the pyplot method
2. when plotting something more advance, use the OO method

In [51]:

```
heart_disease.head()
```

Out[51]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

In [52]:

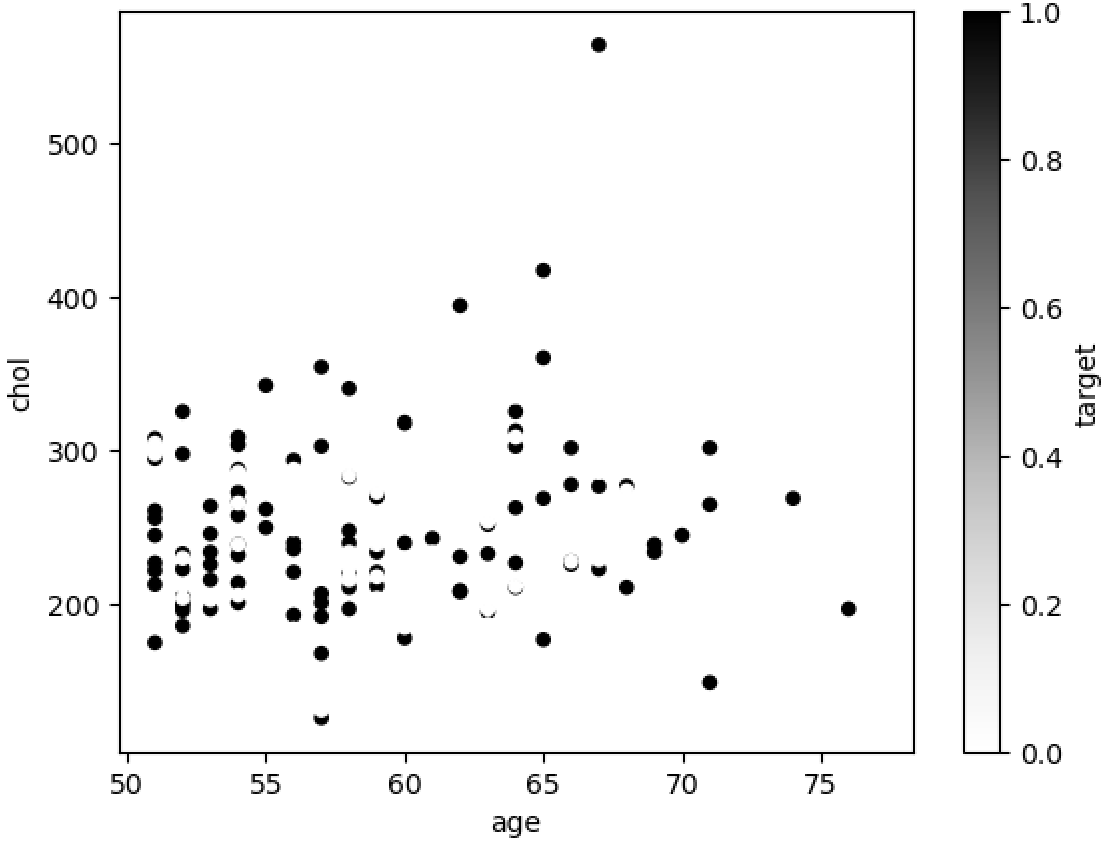
```
over_50 = heart_disease[heart_disease['age'] > 50]
over_50.head()
```

Out[52]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
5	57	1	0	140	192	0	1	148	0	0.4	1	0	1	1
6	56	0	1	140	294	0	0	153	0	1.3	1	0	2	1

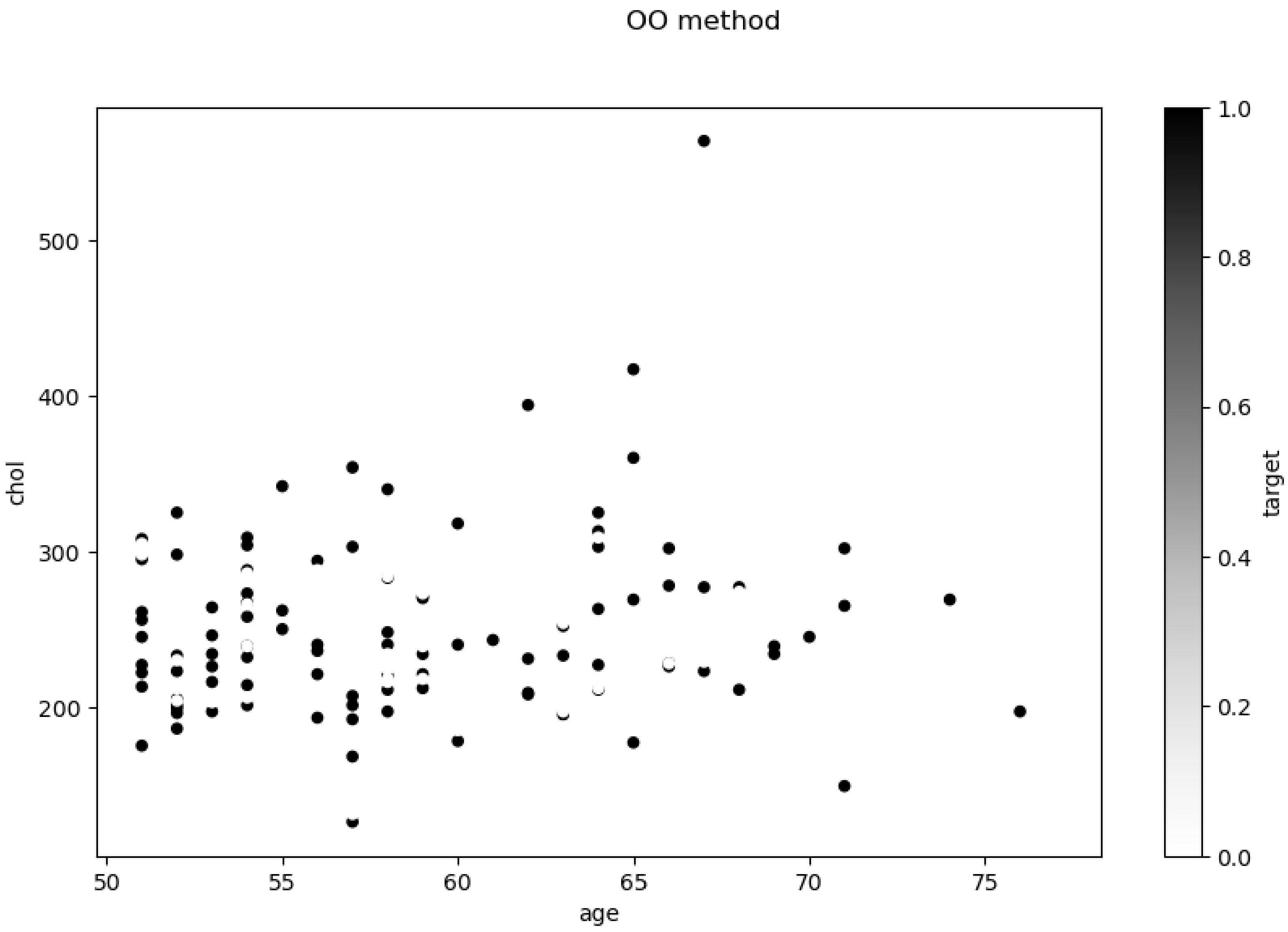
In [53]:

```
# pyplot method
over_50.plot(kind = 'scatter',
             x = 'age',
             y = 'chol',
             c = 'target');
```



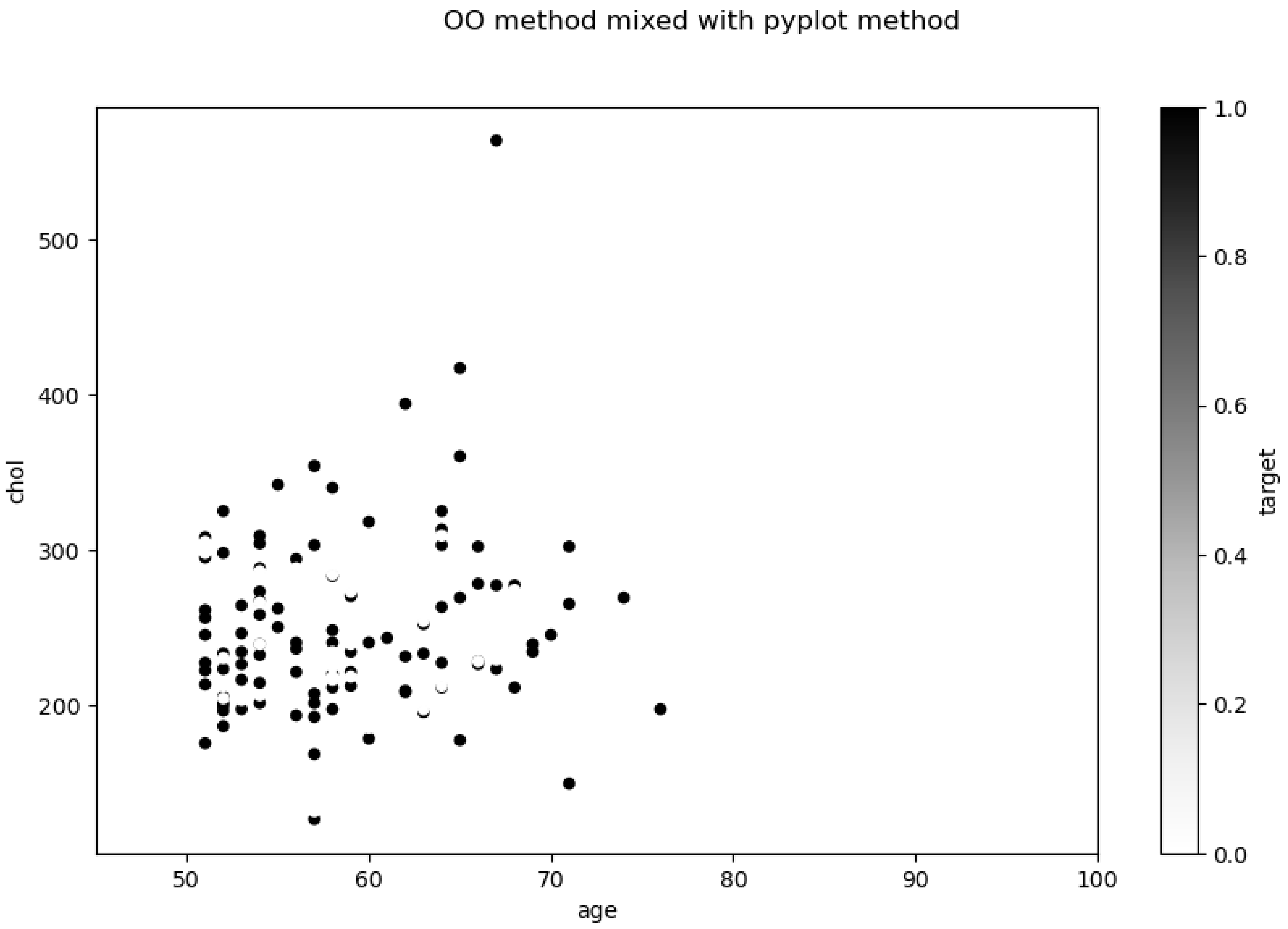
In [54]:

```
# OO method mixed with pyplot method
fig, ax = plt.subplots(figsize = (10, 6))
over_50.plot(kind = 'scatter',
             x = 'age',
             y = 'chol',
             c = 'target',
             ax = ax);
# ax.set_xlim([45, 100])
fig.suptitle('OO method');
```


OO method
chol
500
400
300
200
age
50
55
60
65
70
75
target
1.0
0.8
0.6
0.4
0.2
0.0

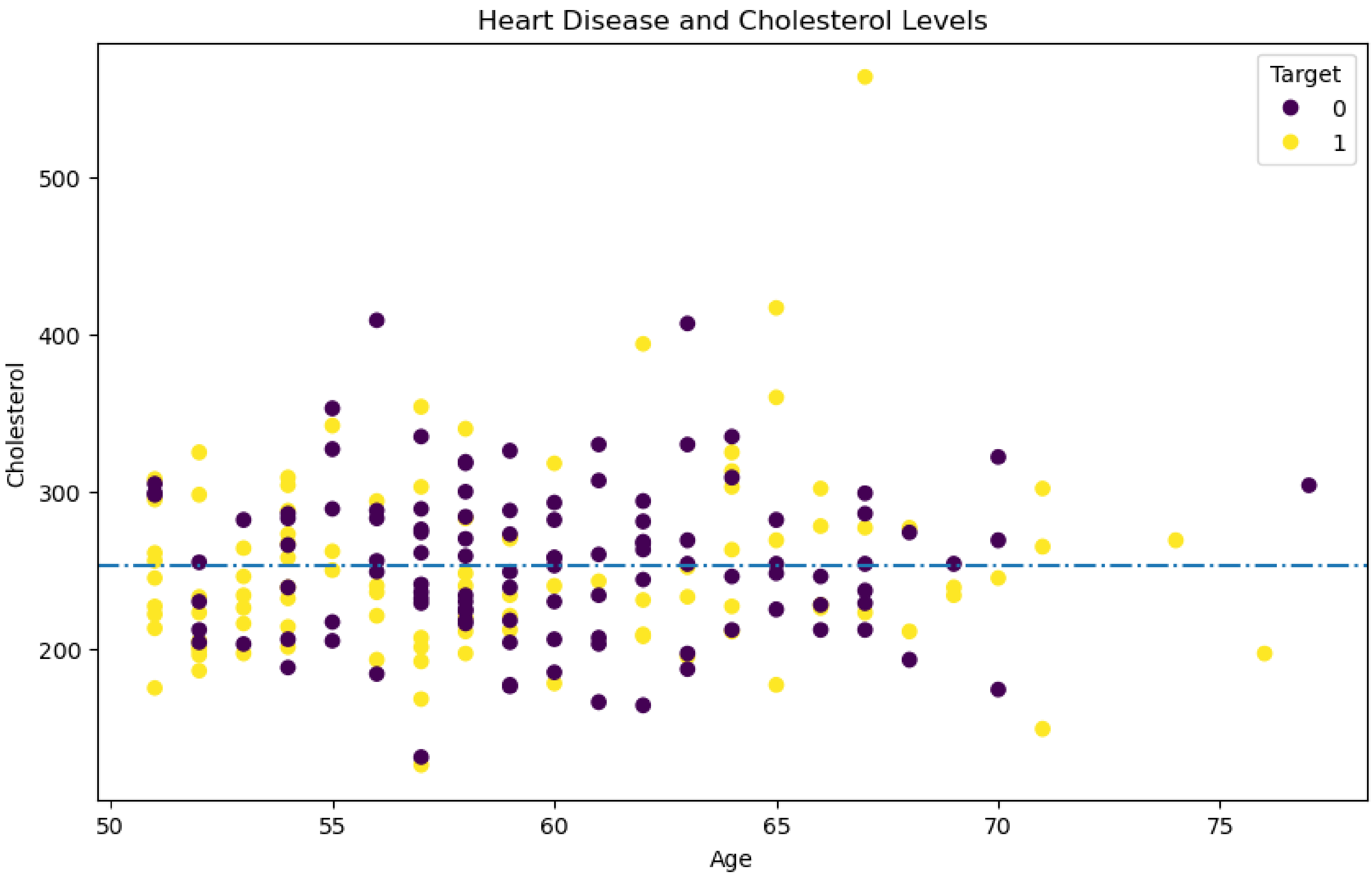
In [55]:

```
fig, ax = plt.subplots(figsize = (10, 6))
over_50.plot(kind = 'scatter',
             x = 'age',
             y = 'chol',
             c = 'target',
             ax = ax);
ax.set_xlim([45, 100])
fig.suptitle('OO method mixed with pyplot method');
```


OO method mixed with pyplot method
chol
500
400
300
200
age
50
60
70
80
90
100
target
1.0
0.8
0.6
0.4
0.2
0.0

In [56]:

```
# OO method from scratch
fig, ax = plt.subplots(figsize = (10, 6))

# plot the data
scatter = ax.scatter(x = over_50['age'],
                     y = over_50['chol'],
                     c = over_50['target'] )

# customize
ax.set(title = 'Heart Disease and Cholesterol Levels',
      xlabel = 'Age',
      ylabel = 'Cholesterol')
# Add legend
ax.legend(*scatter.legend_elements(), title = 'Target')
# add horinzontal line
ax.axhline(over_50['chol'].mean(), linestyle = '-.');
```

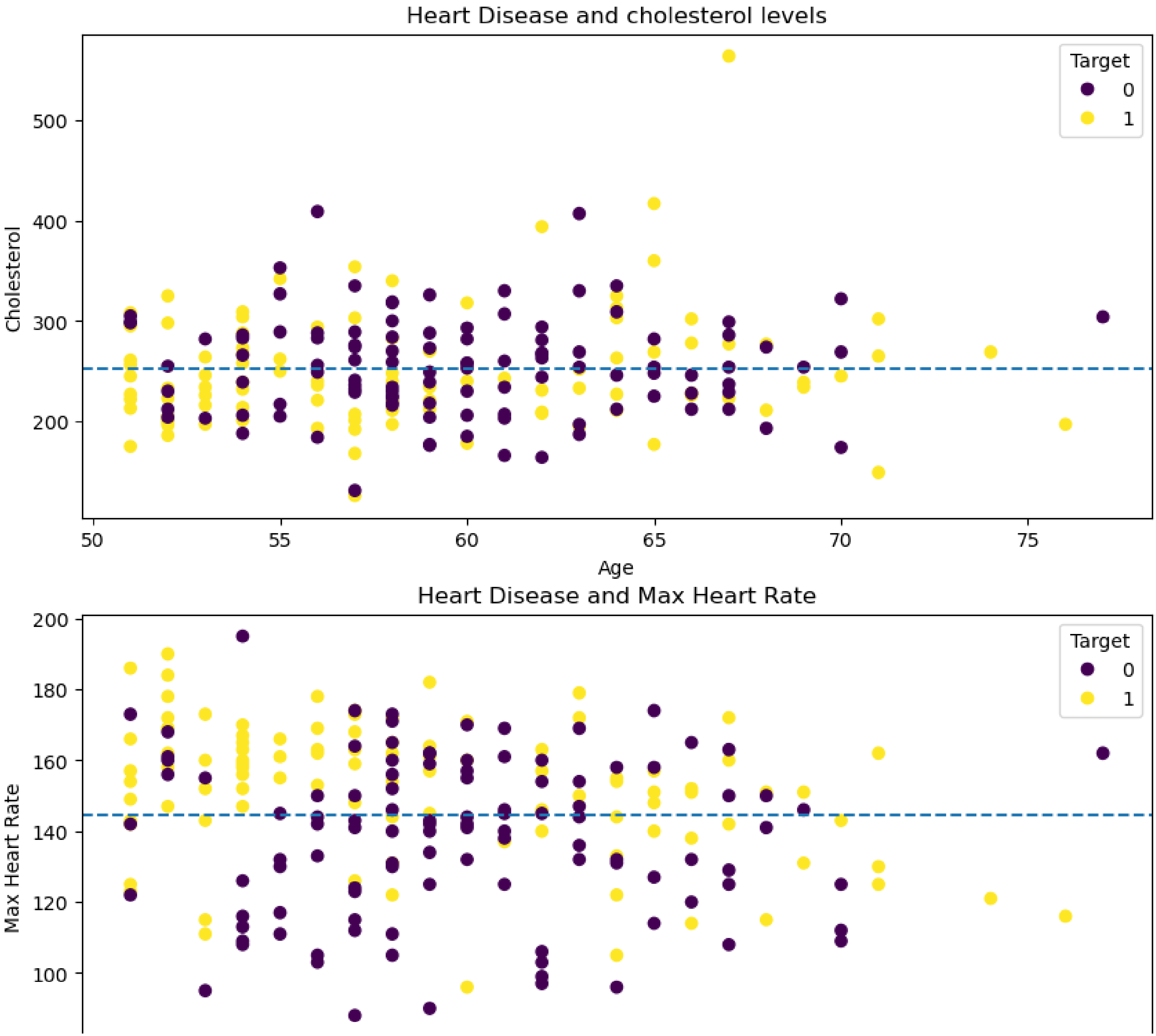

Heart Disease and Cholesterol Levels
Target
● 0
● 1
Cholesterol
500
400
300
200
50
55
60
65
70
75
Age

In [57]:
```
over_50.head()
```

Out[57]:
```
   age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  slope  ca  thal  target
0   63    1   3       145   233    1        0      150      0      2.3      0   0     1       1
3   56    1   1       120   236    0        1      178      0      0.8      2   0     2       1
4   57    0   0       120   354    0        1      163      1      0.6      2   0     2       1
5   57    1   0       140   192    0        1      148      0      0.4      1   0     1       1
6   56    0   1       140   294    0        0      153      0      1.3      1   0     2       1
```

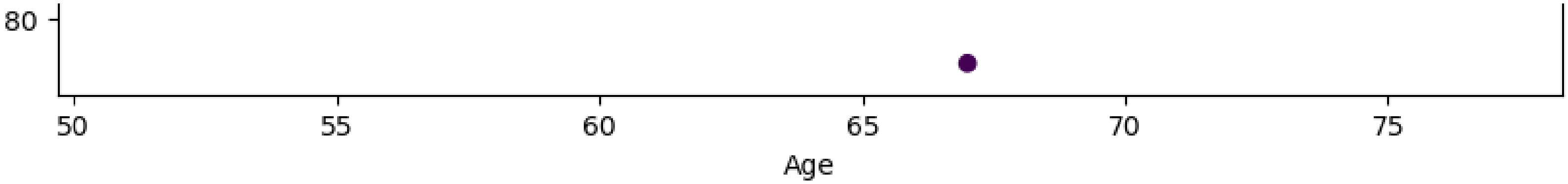
In [58]:

```python
# subplot of chol, age, thalach
fig, (ax0, ax1) = plt.subplots(nrows = 2,
                               ncols = 1,
                               figsize = (10, 10))

# add data to ax0
scatter = ax0.scatter(x = over_50['age'],
                      y = over_50['chol'],
                      c = over_50['target'])

# customize ax0
ax0.set(title = 'Heart Disease and cholesterol levels',
       xlabel = 'Age',
       ylabel = 'Cholesterol');

# add a legend to ax0
ax0.legend(*scatter.legend_elements(), title = 'Target')

# add a meanline
ax0.axhline(y = over_50['chol'].mean(),
           linestyle = '--');

# add data to ax1
scatter = ax1.scatter(x = over_50['age'],
                      y = over_50['thalach'],
                      c = over_50['target'])
# customize ax1
ax1.set(title = 'Heart Disease and Max Heart Rate',
       xlabel = 'Age',
       ylabel = 'Max Heart Rate')

# add legend
ax1.legend(*scatter.legend_elements(), title = 'Target')

# add a meanline
ax1.axhline(y = over_50['thalach'].mean(),
           linestyle = '--');

```



Heart Disease and cholesterol levels

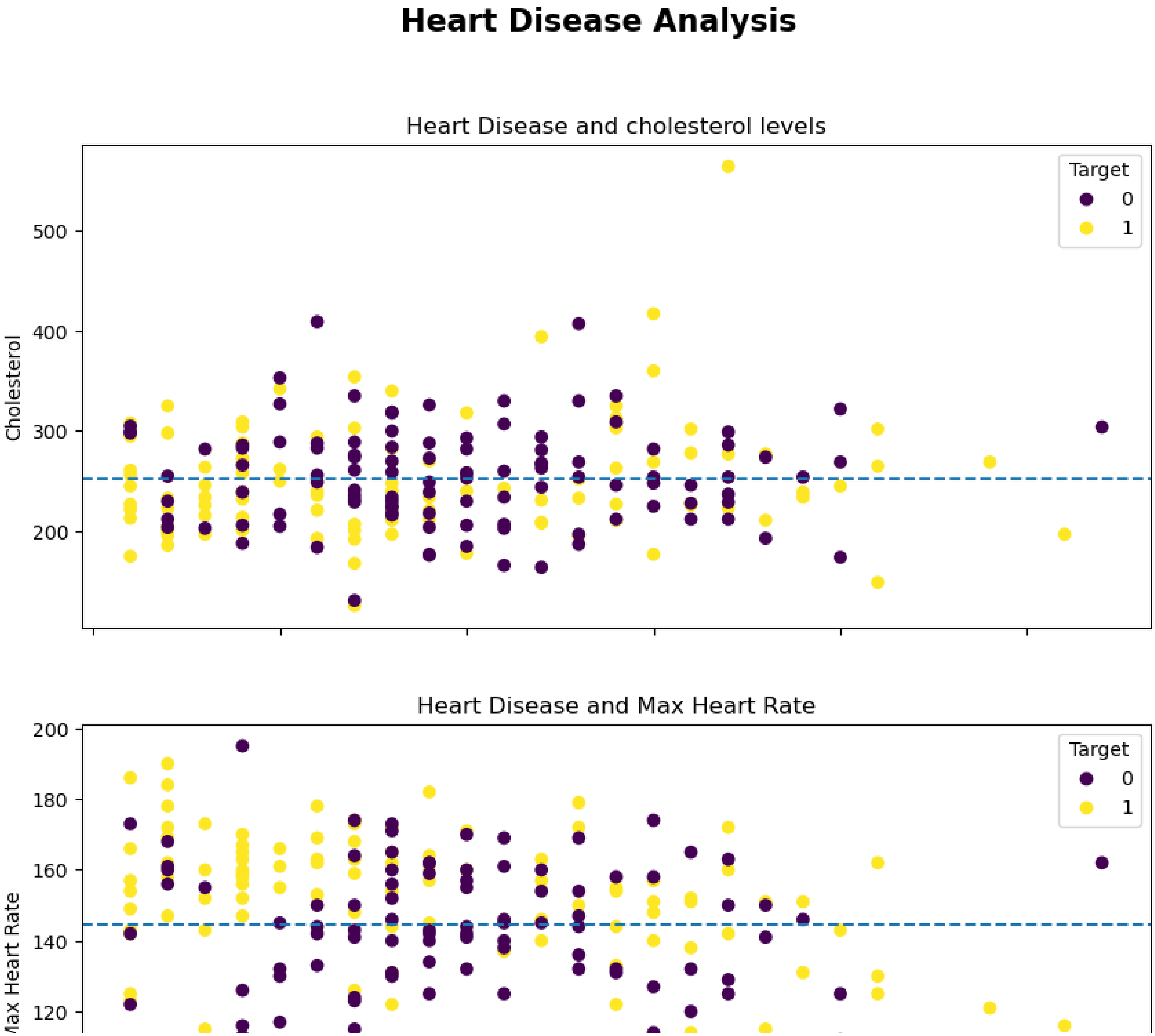
Heart Disease and Max Heart Rate

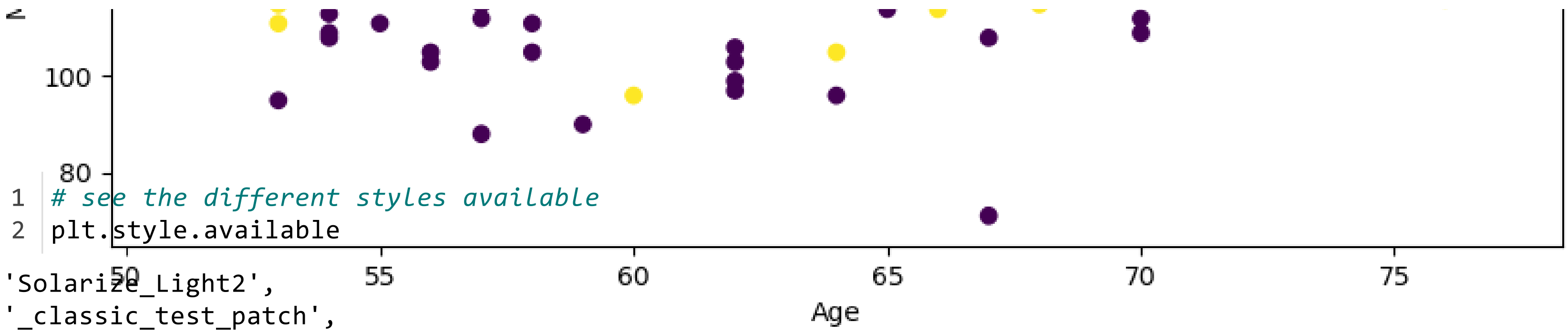

80
50
55
60
65
70
75
Age

In [59]:

```

# subplot of chol, age, thalach
fig, (ax0, ax1) = plt.subplots(nrows = 2,
                               ncols = 1,
                               figsize = (10, 10),
                               sharex = True)

# add data to ax0
scatter = ax0.scatter(x = over_50['age'],
                      y = over_50['chol'],
                      c = over_50['target'])

# customize ax0
ax0.set(title = 'Heart Disease and cholesterol levels',
       ylabel = 'Cholesterol');

# add a legend to ax0
ax0.legend(*scatter.legend_elements(), title = 'Target')

# add a meanline
ax0.axhline(y = over_50['chol'].mean(),
           linestyle = '--');

# add data to ax1
scatter = ax1.scatter(x = over_50['age'],
                      y = over_50['thalach'],
                      c = over_50['target'])
# customize ax1
ax1.set(title = 'Heart Disease and Max Heart Rate',
       xlabel = 'Age',
       ylabel = 'Max Heart Rate')

# add legend
ax1.legend(*scatter.legend_elements(), title = 'Target')

# add a meanline
ax1.axhline(y = over_50['thalach'].mean(),
           linestyle = '--');

# add a title to the figures
```

```
fig.suptitle('Heart Disease Analysis', fontsize = 16, fontweight ='bold');
```


Heart Disease Analysis
Heart Disease and cholesterol levels
Cholesterol
500
400
300
200
Target
● 0
● 1
Heart Disease and Max Heart Rate
Max Heart Rate
200
180
160
140
120
Target
● 0
● 1



In [60]:

```
# see the different styles available
plt.style.available
```

Out[60]:

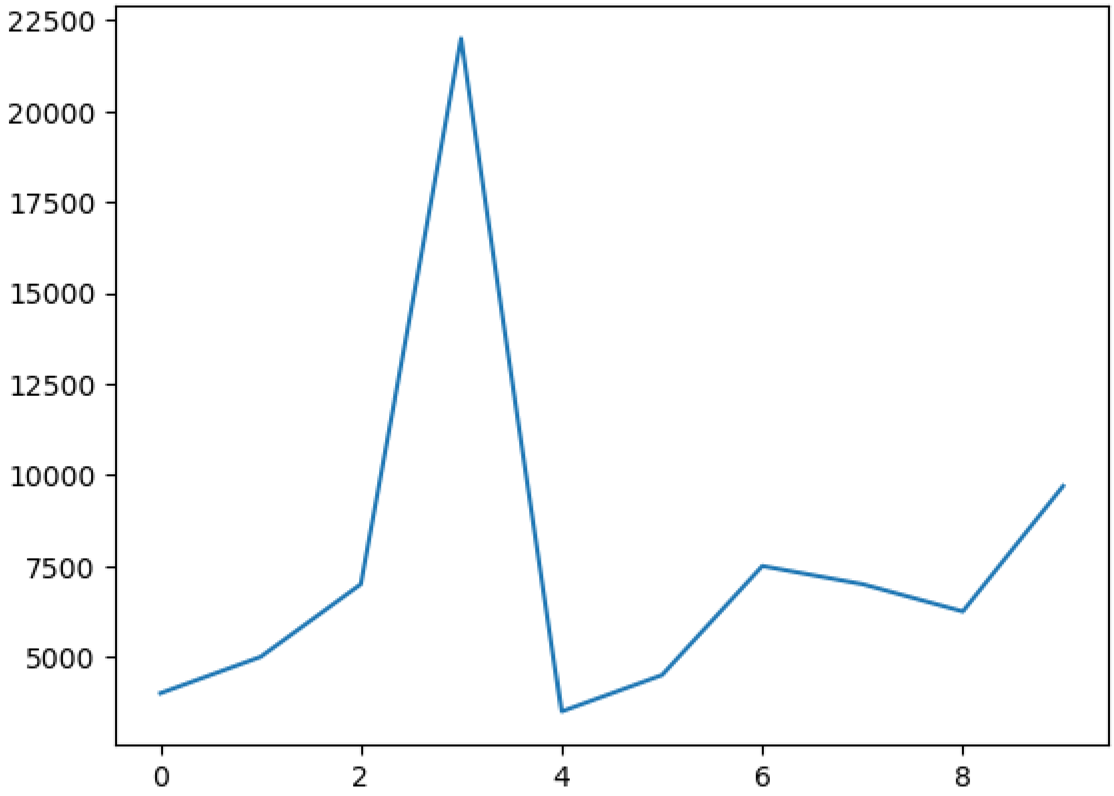
```
['Solarize_Light2',
 '_classic_test_patch',
 '_mpl-gallery',
 '_mpl-gallery-nogrid',
 'bmh',
 'classic',
 'dark_background',
 'fast',
 'fivethirtyeight',
 'ggplot',
 'grayscale',
 'seaborn',
 'seaborn-bright',
 'seaborn-colorblind',
 'seaborn-dark',
 'seaborn-dark-palette',
 'seaborn-darkgrid',
 'seaborn-deep',
 'seaborn-muted',
 'seaborn-notebook',
 'seaborn-paper',
 'seaborn-pastel',
 'seaborn-poster',
 'seaborn-talk',
 'seaborn-ticks',
 'seaborn-white',
 'seaborn-whitegrid',
 'tableau-colorblind10']
```

```
In [61]: car_sales.head()
```

Out[61]:

	Make	Colour	Odometer (KM)	Doors	Price	Sale Date	Total Sales
0	Toyota	White	150043	4	4000	2023-03-07	4000
1	Honda	Red	87899	4	5000	2023-03-08	9000
2	Toyota	Blue	32549	3	7000	2023-03-09	16000
3	BMW	Black	11179	5	22000	2023-03-10	38000
4	Nissan	White	213095	4	3500	2023-03-11	41500

```
In [62]: # default style of plot
         car_sales['Price'].plot();
```

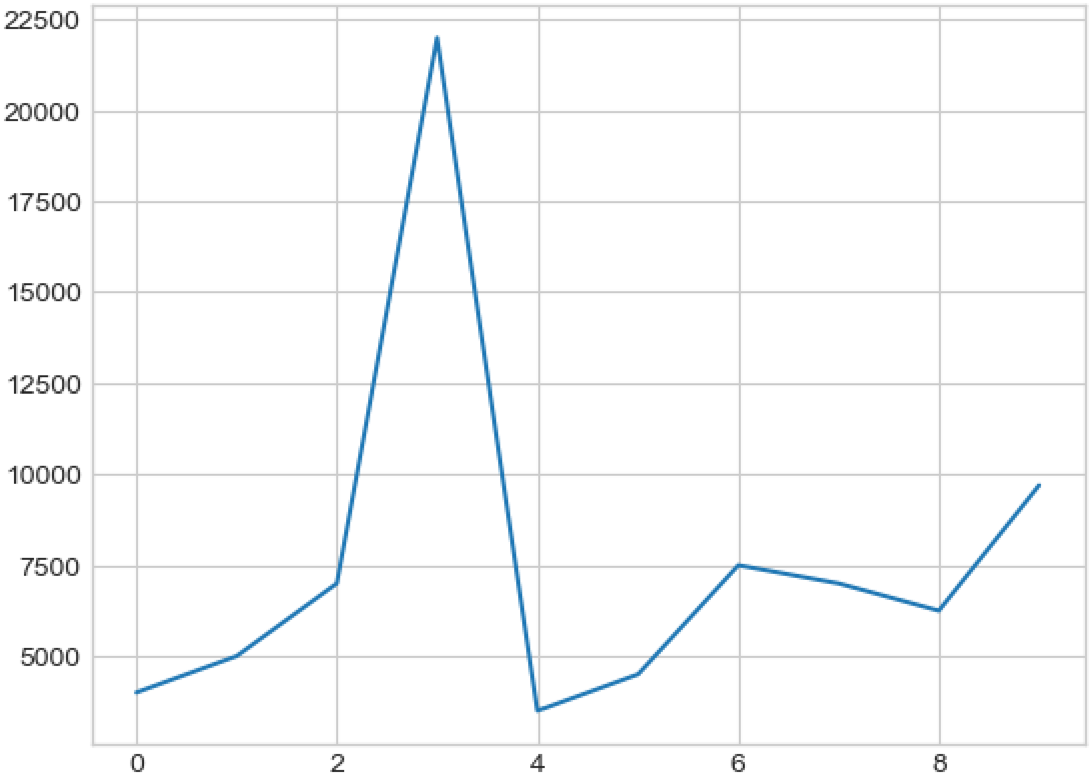


In [63]:

```
plt.style.use('seaborn-whitegrid')
```

In [64]:

```
car_sales['Price'].plot();
```

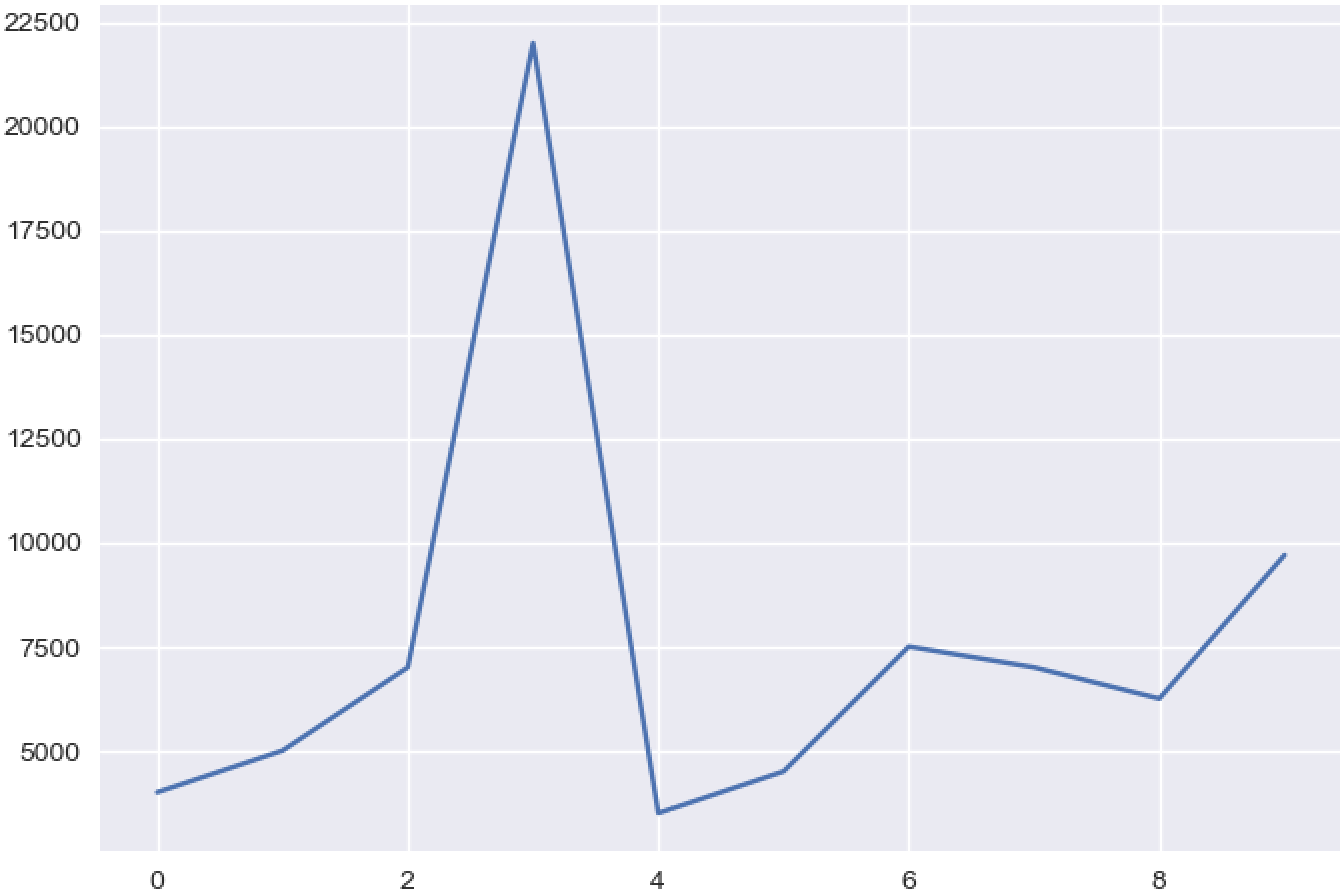


In [65]:

```
plt.style.use('seaborn')
```

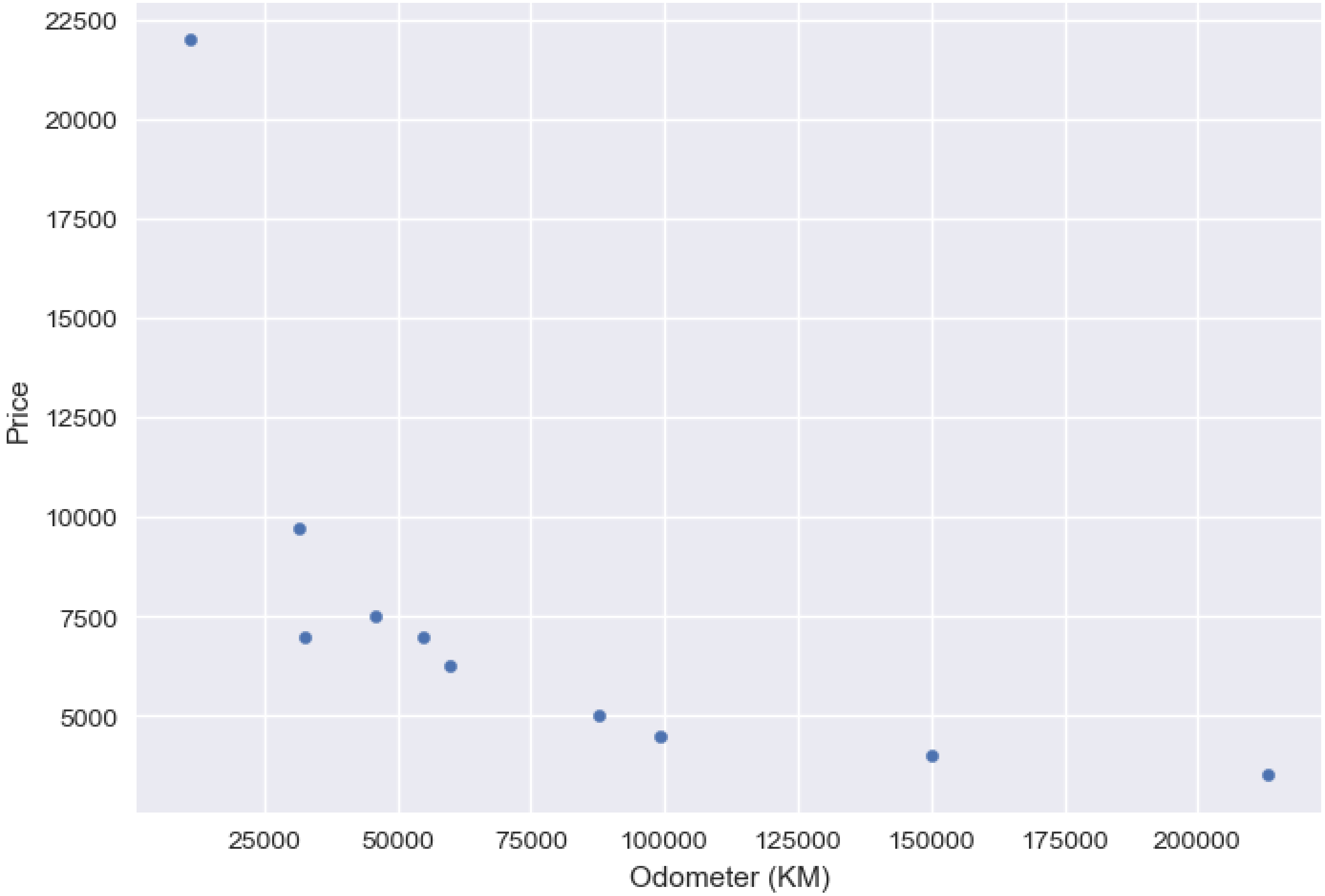
In [66]:

```
car_sales['Price'].plot();
```



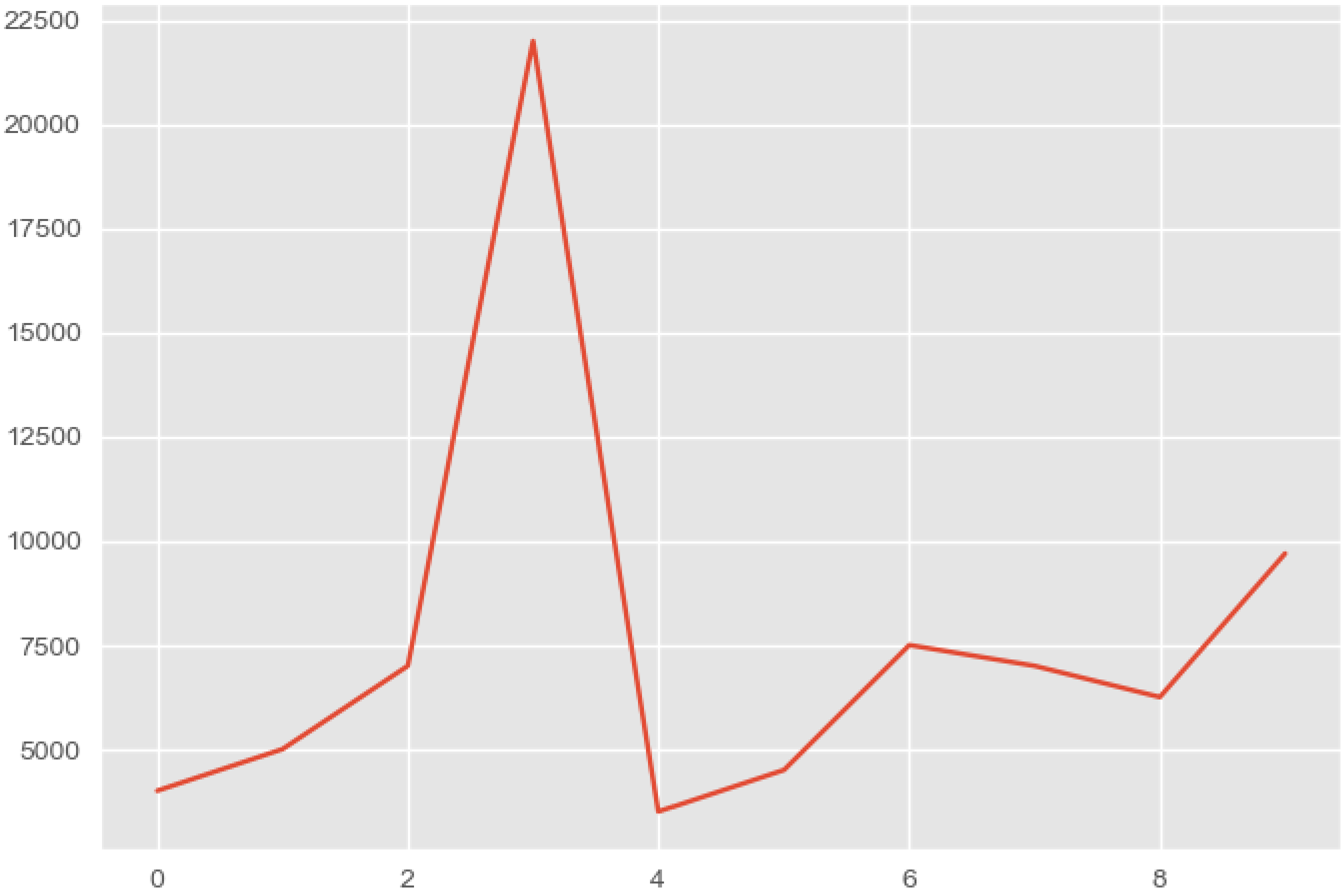
In [67]:

```
car_sales.plot(x = 'Odometer (KM)', y = 'Price', kind = 'scatter');
```



In [68]:

```
plt.style.use('ggplot')
car_sales['Price'].plot();
```



In [69]:

```python
# create some data
x = np.random.randn(10, 4)
x
```

Out[69]:

```
array([[ 0.21856709,  0.40773901, -1.89744613,  0.4625373 ],
       [-0.51108811, -0.42244579, -0.92745166, -1.10178285],
       [ 0.05047383, -1.61504752, -0.06647446, -0.59499287],
       [-1.19509676, -0.03830816,  0.68069208, -0.71982124],
       [ 0.52321047,  0.95253151,  0.5026744 , -0.40421611],
       [ 1.1717757 , -2.00338923, -0.02451604,  1.37006575],
       [ 0.72688768, -0.62142096, -1.50719167, -0.41898985],
       [ 0.78721657,  0.68879791, -0.70190067, -0.99642294],
       [-0.29293654, -0.23317477, -0.56212568,  0.44066704],
       [-0.11005511, -0.69245772,  0.06008774, -2.28787601]])
```

In [70]:

```python
df = pd.DataFrame(x, columns = ['a', 'b', 'c', 'd'])
df
```

Out[70]:

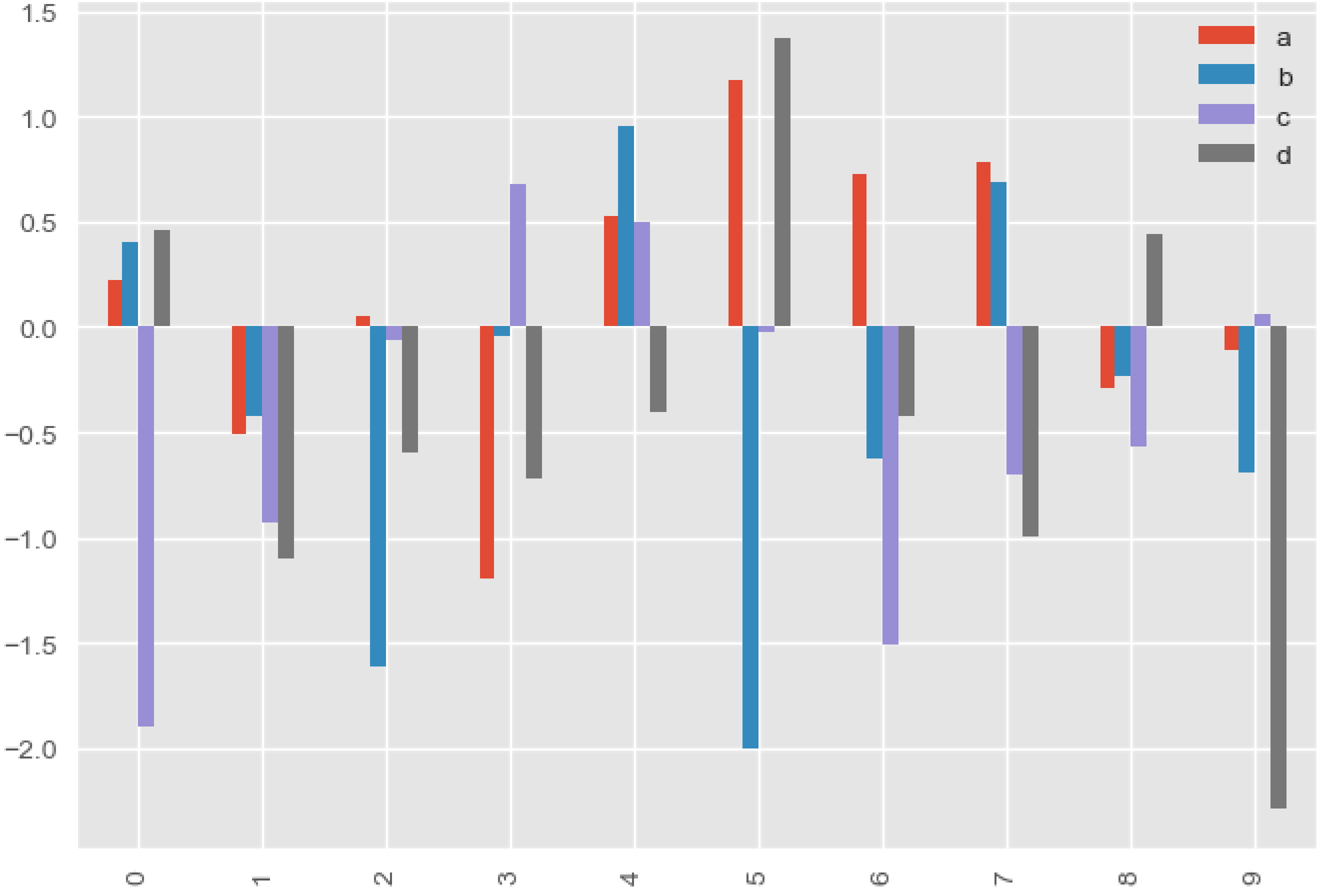
	a	b	c	d
0	0.218567	0.407739	-1.897446	0.462537
1	-0.511088	-0.422446	-0.927452	-1.101783
2	0.050474	-1.615048	-0.066474	-0.594993
3	-1.195097	-0.038308	0.680692	-0.719821
4	0.523210	0.952532	0.502674	-0.404216
5	1.171776	-2.003389	-0.024516	1.370066
6	0.726888	-0.621421	-1.507192	-0.418990
7	0.787217	0.688798	-0.701901	-0.996423
8	-0.292937	-0.233175	-0.562126	0.440667
9	-0.110055	-0.692458	0.060088	-2.287876

In [71]:

```python
ax = df.plot(kind = 'bar')
type(ax)
```

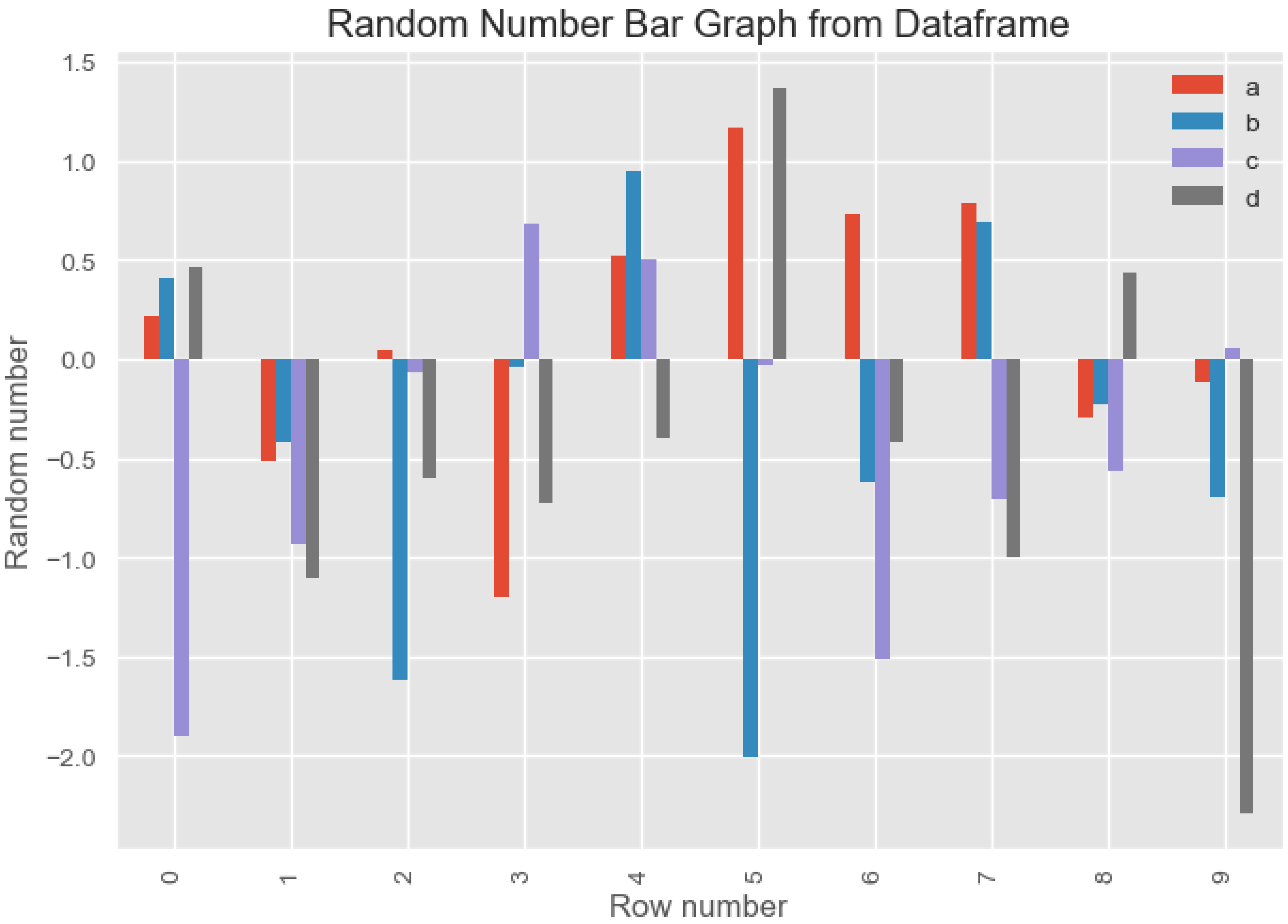
Out[71]:

```
matplotlib.axes._subplots.AxesSubplot
```



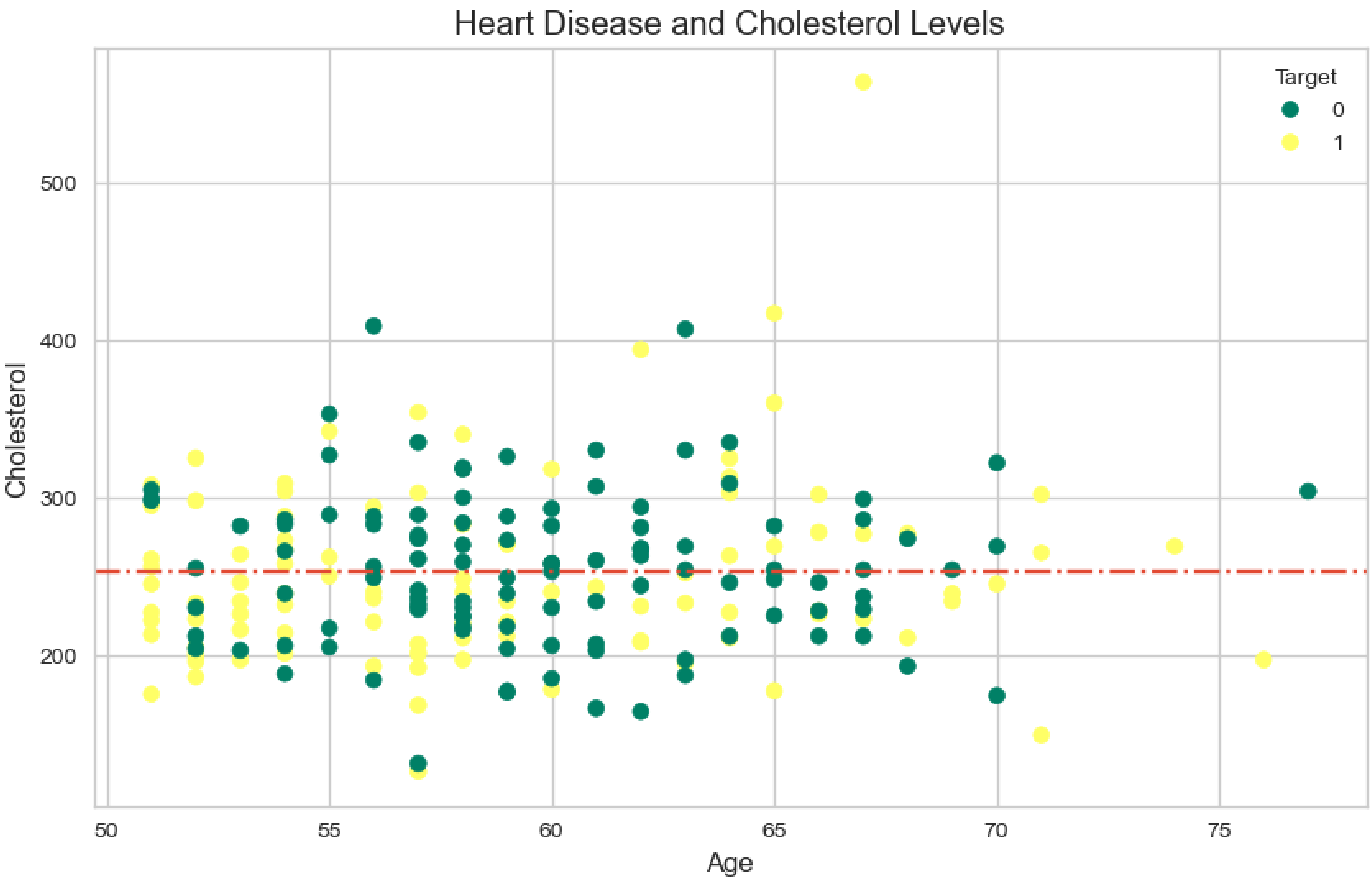
In [72]:

```
# customize our plot with the set() method
ax = df.plot(kind = 'bar')

# add some labels and a title
ax.set(title = 'Random Number Bar Graph from Dataframe',
     xlabel = 'Row number',
      ylabel = 'Random number')

# make the legend visible
ax.legend().set_visible(True)
```



In [73]:

```python
# set the style
plt.style.use('seaborn-whitegrid')

# OO method from scratch
fig, ax = plt.subplots(figsize = (10, 6))

# plot the data
scatter = ax.scatter(x = over_50['age'],
                     y = over_50['chol'],
                     c = over_50['target'],
                     cmap = 'summer') # this chnages the colour scheme

# customize
ax.set(title = 'Heart Disease and Cholesterol Levels',
      xlabel = 'Age',
      ylabel = 'Cholesterol')
# Add legend
ax.legend(*scatter.legend_elements(), title = 'Target')
# add horinzontal line
ax.axhline(over_50['chol'].mean(), linestyle = '-.');
```

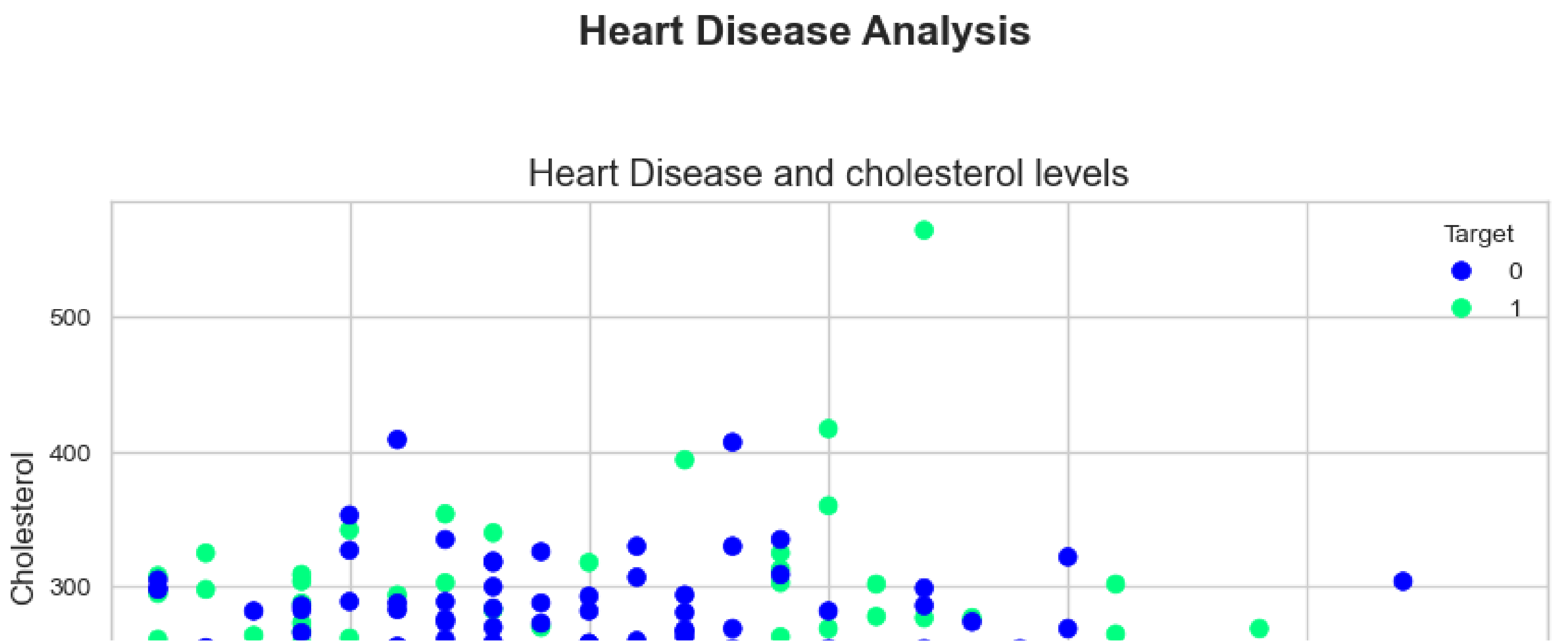


This plot shows infromation about the heart disease dataset...

In [74]:

```
# customizing the y and x axis limitations

# subplot of chol, age, thalach
fig, (ax0, ax1) = plt.subplots(nrows = 2,
                              ncols = 1,
                              figsize = (10, 10),
                              sharex = True)

# add data to ax0
scatter = ax0.scatter(x = over_50['age'],
                     y = over_50['chol'],
                     c = over_50['target'],
                     cmap = 'winter')

# customize ax0
ax0.set(title = 'Heart Disease and cholesterol levels',
       ylabel = 'Cholesterol');
# change the x axis limits
ax0.set_xlim([50, 80])

# add a legend to ax0
ax0.legend(*scatter.legend_elements(), title = 'Target')

# add a meanline
ax0.axhline(y = over_50['chol'].mean(),
           linestyle = '--');

# add data to ax1
scatter = ax1.scatter(x = over_50['age'],
                     y = over_50['thalach'],
                     c = over_50['target'],
                     cmap = 'summer')
# customize ax1
ax1.set(title = 'Heart Disease and Max Heart Rate',
       xlabel = 'Age',
       ylabel = 'Max Heart Rate')
# change ax1 x axis limits
ax1.set_xlim([50, 80])
ax1.set_ylim([60, 200])
# add legend
ax1.legend(*scatter.legend_elements(), title = 'Target')

# add a meanline
```

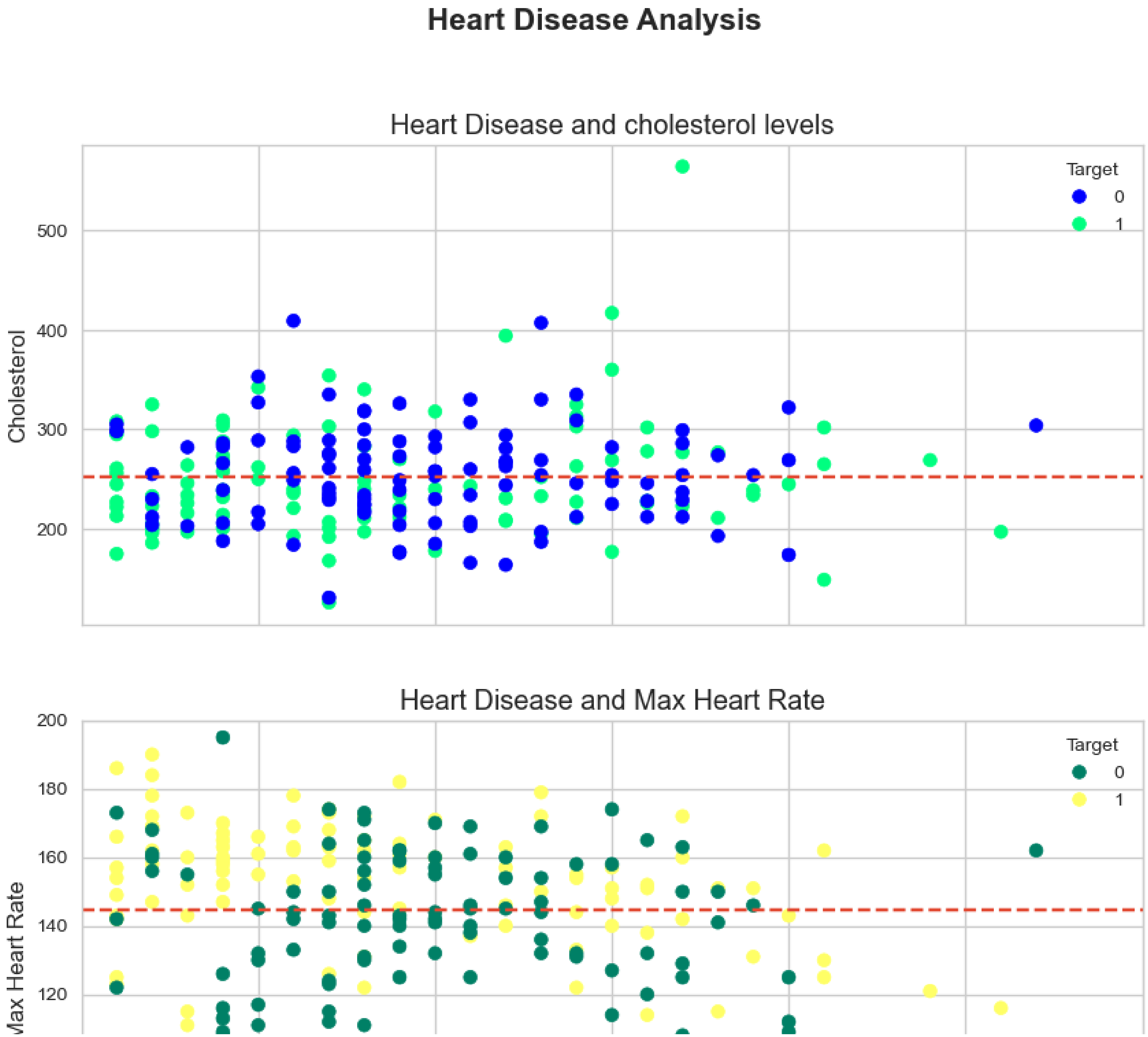
```
ax1.axhline(y = over_50['thalach'].mean(),
           linestyle = '--');

# add a title to the figures
fig.suptitle('Heart Disease Analysis', fontsize = 16, fontweight ='bold');
```

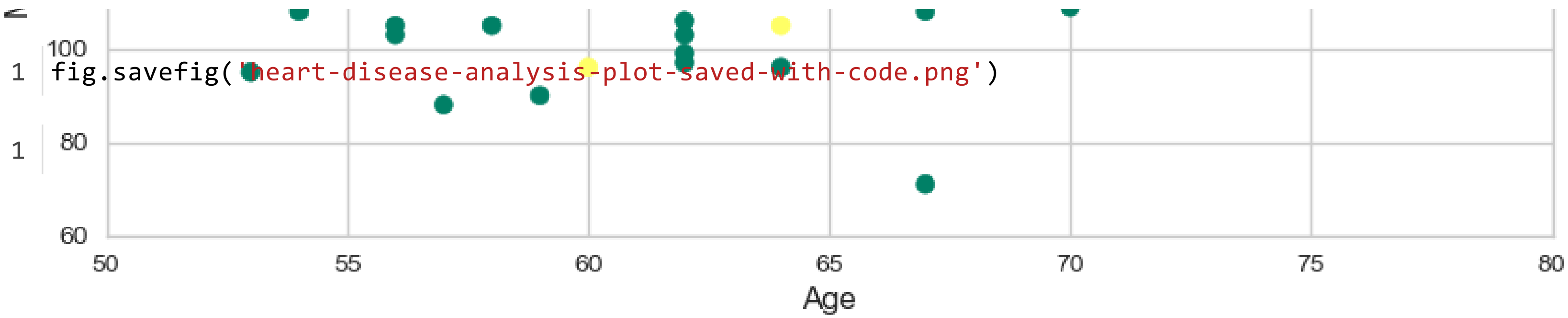


In [75]:

```
fig
```

```
Out[75]:
```


Heart Disease Analysis
Heart Disease and cholesterol levels
Target
● 0
● 1
500
400
300
200
Cholesterol
Heart Disease and Max Heart Rate
Target
● 0
● 1
200
180
160
140
120
Max Heart Rate



In [76]:
```
fig.savefig('heart-disease-analysis-plot-saved-with-code.png')
```

In [ ]:
```

```